ADCS

SPACECRAFT ATTITUDE DETERMINATION AND CONTROL

MICHAEL PALUSZEK

ELSEVIER

ADCS - SPACECRAFT ATTITUDE DETERMINATION AND CONTROL

ADCS - SPACECRAFT ATTITUDE DETERMINATION AND CONTROL

MICHAEL PALUSZEK
Princeton Satellite Systems
Plainsboro, NJ, United States

Front Cover: James Webb Space Telescope
Image Credit: NASA-GSFC, Adriana M. Gutierrez (CI Lab)

Elsevier
Radarweg 29, PO Box 211, 1000 AE Amsterdam, Netherlands
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, United Kingdom
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States

Notices

Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

ISBN: 978-0-323-99915-1

Publisher: Matthew Deans
Acquisitions Editor: Chiara Giglio
Editorial Project Manager: Franchezca A. Cabural
Production Project Manager: Prasanna Kalyanaraman
Cover Designer: Matthew Limbert

Typeset by VTeX

For my wife, Marilyn, and son, Eric.

Contents

This book has a companion website hosting complementary materials. Click on the link to access it: https://www.elsevier.com/books-and-journals/book-companion/9780323999151.

List of examples

List of figures

Biography

Michael Paluszek (1954–)

Michael Paluszek is President of Princeton Satellite Systems, Inc. (PSS) in Plainsboro, New Jersey. Mr. Paluszek founded PSS in 1992 to provide aerospace consulting services. Projects at PSS include optical navigation for spacecraft, agent-based software for formation flying, the TDRS H, I, J momentum-management systems and the Swedish Space Corporation's Prisma Rendezvous Robots formation flying experiment.

He is working with the Princeton Plasma Physics Laboratory on a compact nuclear-fusion reactor for terrestrial energy generation and space power and propulsion. He is also working on Brayton Cycle heat engines for space applications.

Prior to founding PSS, Mr. Paluszek was an engineer at General Electric (GE) Astro Space in East Windsor, NJ. At GE, he designed an early version of the Global Geospace Sciences Polar despun-platform control system and led the design of the GPS IIR attitude control system, the Inmarsat-3 attitude control systems and the Mars Observer Delta-V control system. Mr. Paluszek flew communication satellites on over twelve satellite launches, including the GSTAR III recovery, the first transfer of a satellite to an operational orbit using electric thrusters.

At Draper Laboratory, Mr. Paluszek worked on the Space Shuttle, Space Station, and submarine navigation. His Space Station work included the design of control-moment gyro-based control systems for attitude control.

He worked at MIT on microthrusters under Professor Manuel Martinez-Sanchez. He also worked with Professor Rene Miller on aeronautics research.

Mr. Paluszek received his Bachelor's degree in Electrical Engineering, his Master's degree of Aeronautics and Astronautics, and his Engineer of Aeronautics and Astronautics degree from the Massachusetts Institute of Technology. He is the author of numerous technical papers and has over a dozen US Patents. Mr. Paluszek is the author of *MATLAB Recipes*, *MATLAB Machine Learning*, *MATLAB Machine Learning Recipes: A Problem-Solution Approach*, *Practical MATLAB Deep Learning Projects*, *MATLAB Recipes Revised Edition*, and *Practical MATLAB Deep Learning* published by Apress.

Preface

ADCS - Spacecraft Attitude Determination and Control was written to provide an accessible book on designing spacecraft control systems. The idea was to mix theory and practice so that the reader can immediately implement the concepts discussed in this book. For that reason, MATLAB® code is given for most key elements in the book. The code can easily be converted to Python or other languages at the preference of the reader.

ADCS is a huge field. The first control systems were implemented in the 1960s prior to the introduction of digital flight computers. As a consequence, there is a vast literature on the subject. In this book, I have tried to cover the full range of approaches, from passive control all the way to Artificial Intelligence-based approaches. In this way, the reader will not have to follow in the pattern of designing a new system like the last system they did, just because it is familiar. I have tried to be compact in the presentation so you will not find long derivations or theorems and proofs. In some cases just enough information is provided to give you a starting point for further research. Most of the book is based on my experiences in spacecraft control. I do discuss other spacecraft for which there are good references in the literature.

I have been working in the area of spacecraft control for forty years, starting with learning about the Space Shuttle Orbiter at the Charles Stark Draper Laboratory, even writing code in HAL/S the Space Shuttle programming language, through working on numerous control-systems designs at GE Astro Space, and then working on many new concepts at my company, Princeton Satellite Systems. Back when I started Princeton Satellite Systems, there were only a couple of dozen engineers who had designed spacecraft control systems. Nowadays, starting with the introduction of CubeSats through the massive commercial constellations, there are hundreds, if not thousands of experienced engineers. Even those with experience will hopefully find this book valuable, if only to provide other approaches to the problems they need to solve. That said, the approaches in this book are not the only way to handle the problems and the reader should not feel constrained in any way by the ideas presented in the book.

Part 1 of the book outlines ADCS theory. Many short examples are given to show how the theory works in practice. Classes of spacecraft are presented to motivate the theory and give the reader a way to quickly use the material. The book tries to make chapters self-contained so the reader will see some repetition of material. This is intentional.

Part 2 of the book gives examples from industry. Examples range from CubeSats to solar sails. The book does not necessarily replicate the control systems that are actually

used, instead it shows the process to design an ADCS system that meets the general requirements of the mission.

The appendices present background theory in control, statistics, math, and other topics that are helpful to understand the material in the book.

The book does not feature any problems at the end of the chapters. Instead, it provides MATLAB source code for all of the examples. Once you have set up your paths to the supporting function folder, you can run all of the code. The code includes short snippets to explain a point in the text and complete simulations. The provided code can be used as a starting point for your work. For experienced ACS designers, the book may provide alternative approaches to ACS design. For other subsystem designers, it gives a thorough overview of ACS design so that you will know what the ACS people are doing.

This book was written over the course of many years. Some of the material is in the reference book for my company's software products. I have incorporated material that I use in teaching an MIT Attitude Control course. The newest material, on Artificial Intelligence, has its origins in 6.034 Artificial Intelligence taught by Professor Patrick Winston at MIT in 1973, and various explorations in AI on different spacecraft projects.

I hope you enjoy the book. Please contact me with your questions, suggestions, or corrections!

Michael Paluszek

Acknowledgments

This book is based on the experience of the author in spacecraft attitude and orbit control. This started with work on the NASA Space Shuttle at the Charles Stark Draper Laboratory and continued with work on numerous spacecraft at RCA Astro Electronics (which became GE Astro Space and is now part of Lockheed Martin). Most of the recent work discussed in this book relates to work done by Princeton Satellite Systems on numerous Air Force, NASA, and commercial contracts.

The author would like to thank the many colleagues with whom he has worked over the past 40 years who in some way influenced the work in this book. This includes collaborators on projects and managers of projects on which Princeton Satellite Systems worked. In particular, I would like to thank current and former employees of Princeton Satellite Systems, Stephanie Thomas, Derek Surka, Joseph Mueller, Wendy Sullivan, Charles Swanson, Christopher Galea, and Eric Ham. Stephanie Thomas contributed to much of the code in this book. She also was responsible for much of the theory for star cameras and solar sails. I would like to thank the dozens of interns who worked at the company. I would like to thank Princeton Satellite Systems for the use of the underlying code that is used in many of the examples.

I would like to thank Professor Manuel Martinez-Sanchez, who was my thesis advisor at MIT for both my undergraduate and graduate degrees, Mr. Paul Zetocha, manager of numerous Princeton Satellite Systems contracts at the US Air Force Research Laboratory (AFRL), Joel Storch premier dynamicist at the Charles Stark Draper Laboratory, Dr. Barbara Sorensen of the US Air Force, Björn Jakobsson of the Swedish Space Corporation with whom I worked on the PRISMA project, Mr. Douglas Freesland who awarded us the Indostar-1 attitude control system contract, Dr. Alfred Ng of the Canadian Space Agency, Mr. Douglas Bender of Boeing with whom I worked on TDRS H, I, and J, Mr. Rich Burns of NASA with whom I collaborated on TechSat-21, Dr. Russell Carpenter of NASA who managed one of our formation flying contracts, Dr. Neil Goodzeit, Dr. James Swale, Dr. Carl Hubert, and Dr. Ludwig Muhlfelder of Lockheed Martin, Bruce Campbell, and Andy Heaton of NASA with whom I worked on solar sails, Dr. Robert McKillip of Continuum Dynamics who collaborated with me on fault detection, and Mr. Christian Phillippe of ESTEC with whom I did solar-sail work. I would like to thank Professors Daniel Hastings, Olivier de Weck, and Kerri Cahoy of MIT who hired me to teach attitude control systems at MIT. Teaching the course at MIT has led to many improvements in this book. I would like to thank Michael Brown, a student in my MIT class, for reviewing an early draft of the book. His suggestions were very valuable.

This list is not necessarily complete and I apologize to anyone I have overlooked!

CHAPTER 1

Introduction

1.1. Overview of the book

Attitude Determination and Control Systems are the software, hardware, and algorithms that keep a spacecraft pointed in the desired direction; that is they control the spacecraft's attitude or orientation. Geosynchronous communications satellites need to point their antennas precisely at the Earth to transmit data, voice, television, and data, that produce their revenue stream. A spacecraft docking with the International Space Station needs to keep its docking port aligned with the counterpart on the station. A lunar lander must align its thruster with the desired thrust vector while making sure that its sensors are oriented to receive the data needed for the landing. While often the spacecraft is pointing at one target and rotating slowly with respect to an inertial reference frame, sometimes the control system must rotate the spacecraft quickly.

An attitude control system needs data measuring its orientation with respect to a reference. For a docking spacecraft, such as the Russian Progress, that is the International Space Station. For an Earth-pointing satellite such as a SpaceX Starlink, it is the Earth. For a space telescope, like the Hubble or James Webb, it is an inertial reference, usually the starfield. Often a direct measurement, such as of the attitude with respect to the Earth, is not available with sufficient precision so another measurement, such as the orientation with respect to the star field, is used. The relative orientation with respect to the Earth must be inferred from knowledge of the Earth's rotation with respect to the field of stars. Advanced control systems may be able to identify a target on a planetary or lunar surface and track that target, or in the case of landers, avoid something it does not want to hit.

Once the system has the orientation information, it must compute torques to maintain the orientation or to reorient the spacecraft. This is done with the control actuators such as momentum-exchange devices, thrusters, or magnetic torquers. The disturbances, which perturb the orientation, are generally very small, on the order of micro-N m, but they are applied over long periods, and compensation for their effects is required. They are also a function of the time of year and the orbital position of the spacecraft, further complicating the problem. The actuators, whether thrusters, magnetic torquers, or reaction wheels, tend to be nonlinear and nonideal. The control algorithms couple the measurements to the actuators. While the controller is often just a regulator, which is a controller that maintains the desired orientation, the control algorithms sometimes have to respond to ground commands from operators or the crew onboard.

The dynamics and kinematics of a spacecraft are inherently nonlinear. For small angular rates and small angles from the nominal orientation it is usually possible to

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00011-5

linearize the equations of motion. For an inertially pointing spacecraft, this is a fixed inertial direction. For a planet pointer, this is in a rotating coordinate frame. This lets the designer apply linear control theory.

This book provides an introduction to attitude dynamics, estimation, and control for spacecraft. This book gives you the information needed to solve problems for a wide variety of spacecraft. It sets the design of attitude control systems in the overall spacecraft context. Control systems do not operate in a vacuum and must interact with the other subsystems of the spacecraft, including power, thermal, payload, propulsion, mechanisms, and communications. An attitude control system designer must have a strong background in all of the spacecraft systems since the orientation of the spacecraft affects every subsystem.

The book is organized with the chapters in order of how you design a control system for a spacecraft. Background material such as control theory, actuator and sensor details, and probability and statistics is located in the appendices. The reader can consult the appendices as needed. The book is designed to not require any technical background beyond high-school or freshman-college physics and does not require additional references. MATLAB® software is provided that generates all of the plots in the book. The book tries to avoid "black boxes." The reader is encouraged to read through all of the code that she or he uses.

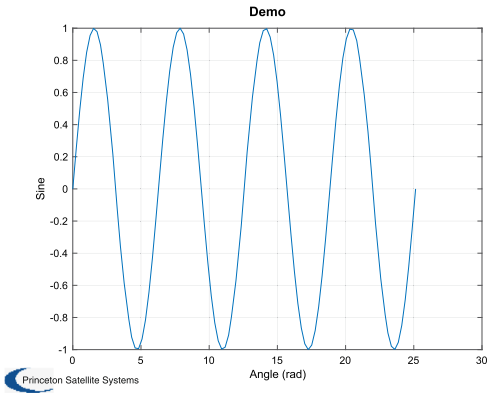
After the design chapters, the next chapters give detailed designs of real spacecraft based on the author's design experience. These include a geosynchronous communications satellite, broken into chapters on momentum-bias control and spinning-spacecraft control. Another chapter discusses Sun-nadir pointing control, such as is used on GPS IIR. One chapter covers passive control that is popular for CubeSats and nanosatellites. Another chapter is on lander control. The last chapter is on the James Webb Space Telescope. We provide a "point" solution that is not necessarily the one the James Webb designers used. A chapter on spacecraft simulations is included along with a chapter on spacecraft operations.

The book also includes a chapter on the history of spacecraft. The chapter includes the evolution of spacecraft and satellite design. Most chapters start with a "Space Story" from the author's experiences in the space business.

The book includes example code written in MATLAB. Example 1.1 shows a typical example code. In this case, we plot a sine wave. All of the functions used in the example are available for download if they are not built-in MATLAB functions. Since you receive all of the source code you can translate them into your language of choice. For example, going from MATLAB to Python is particularly easy.

You run the script SetBoothPaths to set the paths to all of the chapter code and the supporting functions. The script can also run all the scripts and list all of the scripts and functions.

```
a = linspace(0,8*pi);

Plot2D(a,sin(a),'Angle (rad)','Sine','Demo');
```



Example 1.1: A sine wave computed in MATLAB. Code examples in this book will be in this format.

For most of the examples, standard MATLAB will suffice. For the Machine Intelligence chapter, the book uses the Deep Learning and Reinforcement Learning toolboxes. These can be purchased as add-ons to basic MATLAB.

The goal of the book is to give you everything you need to be a designer of attitude control systems. Whether you are a high-school student building a CubeSat or an engineer at an aerospace company, this book has something for you. For professors teaching spacecraft control, the book includes three course outlines for different length courses. The author uses this book to teach attitude control at MIT and is happy to answer questions from faculty teaching this very exciting field.

I hope that you enjoy this book!

1.2. Types of spacecraft

Spacecraft come in every imaginable shape and size. Many nations and organizations, including China, Russia, India, the European Space Agency, Japan, and Korea have launched sophisticated spacecraft. Fig. 1.1 shows a selection of NASA spacecraft, both operational and proposed.

GGS Polar is a bias-momentum spacecraft with a despun platform. The spun part has balance masses on lead screws to keep the center-of-mass on the spin-axis. Cassini is a Saturn-orbiting spacecraft. Its control system is called the Cassini Attitude and Articulation Control Subsystem (AACS) since it controls the articulation of many booms. Cassini is three-axis stabilized using either reaction wheels or thrusters. Cassini also carried the Huygens probe that landed on Titan. Mars Observer was 3-axis stabilized and used reaction wheels and thrusters. Mars Observer was the first NASA spacecraft with reaction wheels. It exploded during its Mars insertion burn. Tracking Data Relay Satellites (TDRS) H, I, and J are NASA geosynchronous communications satellites based on

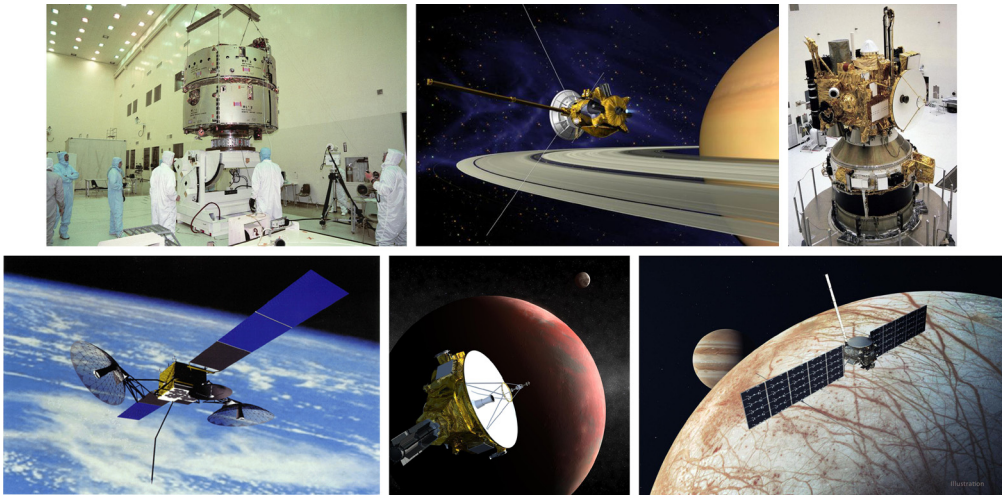


Figure 1.1 NASA spacecraft. From upper left, clockwise. GGS Wind, Cassini, Mars Observer, TDRS H, New Horizons, and Europa Clipper. The author worked on GGS Polar, Mars Observer, and TDRS H, I, J. All images are courtesy of NASA.

the Hughes (now Boeing) HS 601 bus. Bus is a term used for the spacecraft or satellite body where most of the components reside. They use a tilt wheel with two tilt actuators for yaw and roll control. The momentum wheel that is located on the tilt platform is used for pitch control. The solar wings are used for momentum unloading. New Horizons is a NASA interplanetary spacecraft. It flew past Pluto and is still in operation. It is 3-axis stabilized using thrusters. Europa Clipper is a new spacecraft. It has 22 4-N thrusters for control. One of the biggest challenges for Europa Clipper is that it needs to operate in a very high radiation environment.

Fig. 1.2 shows the European Space Agency BepiColombo Mercury Planetary Orbiter at Mercury.

The ACS for BepiColombo is designed with considerable redundancy. It has the following attitude control hardware suite [1,2]:

1. Three star trackers;
2. Two Inertial Measurement Units (IMU);
3. Four fine Sun sensors;
4. Four reaction wheels;
5. Eight 22-N monopropellant hydrazine thrusters for orbit control;
6. Eight 10-N monopropellant hydrazine thrusters for control and momentum unloading.

An interesting feature is the use of a heuristic version of simplex (linear programming) for thruster selection.

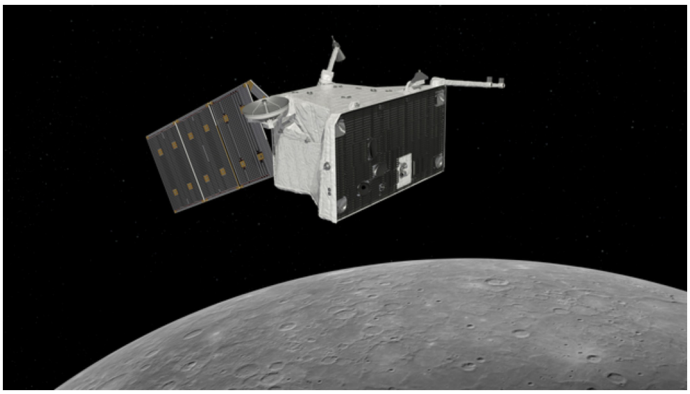


Figure 1.2 European Space Agency BepiColombo Mercury Planetary Orbiter. Copyright: spacecraft: ESA/ATG medialab; Mercury: NASA/Johns Hopkins University Applied Physics Laboratory/Carnegie Institution of Washington.

Fig. 1.3 shows the European Space Agency BepiColombo Mercury Magnetospheric Orbiter at Mercury [3].

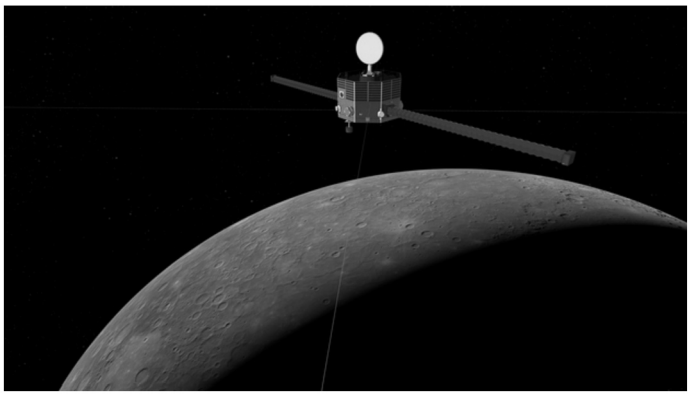


Figure 1.3 European Space Agency BepiColombo Mercury Magnetospheric Orbiter. Copyright: spacecraft: ESA/ATG medialab; Mercury: NASA/Johns Hopkins University Applied Physics Laboratory/Carnegie Institution of Washington.

This spacecraft has a high level of redundancy. This is typical for high-value missions. The Mercury Magnetospheric Orbiter is a spin-stabilized spacecraft.

Table 1.1 gives a list of spacecraft types and their dynamic characteristics.

A gyrostat is a "rigid body" with symmetric rotors.

Table 1.1 Spacecraft examples. CMGs are control-moment gyros. The table is from the simplest attitude control systems design to the most complex design.

Type	Dynamics	Control	Examples
Passive	Rigid body	Gravity gradient, dampers, torquers	GEOS 2.
Spinner	Rigid body	Dampers	Global Geospace Science Wind spacecraft.
Three-axis	Rigid body	Thrusters	Pioneer, Space Shuttle Orbiter, Voyager.
Dual Spin	Gyrostat	Momentum wheel, torquers, thrusters	GE Series 3000-5000, Inmarsat 3, Indostar, BSat, Hughes HS376.
Zero momentum	Gyrostat	Reaction wheels, thrusters, torquers	GPS IIR, DMSP, Tiros, Hubble.
Zero momentum, fast reorientation	Gyrostat	Control-moment gyros, thrusters, torquers	Military Earth-observation satellites.
Zero momentum, with slosh	Gyrostat with pendulum	Reaction wheels, thrusters	Mars Observer.
Multibody gyrostat	Gyrostat with appendages	Pivoted momentum wheel	TDRS H, I, J.
Articulated	Multibody	CMGs	International Space Station (ISS).

TDRS is the NASA Tracking Data Relay Satellite, GPS is the Global Positioning Spacecraft, DMSP is the Defense Meteorological Satellite Program and GEOS is the Geodetic Earth Orbiting Satellite.

Control-moment gyros (CMG)s, are actuators (and not to be confused with gyros, which are sensors). A CMG can be one or two axes. A gyrostat is a type of multibody spacecraft composed of a rigid core and a set of rigid symmetric rotors. A multibody is a spacecraft with points of articulation, much like a terrestrial robot. Joints can be rotational or prismatic (translational). In a gyrostat, rotation does not change the inertia of the spacecraft. In a multibody spacecraft, it does. A gyrostat is a type of multibody spacecraft. More general multibody spacecraft have parts that can translate and rotate with respect to the core spacecraft and the rotations are not necessarily about their center-of-masses. The Space Shuttle Orbiter Remote Manipulator system is an example of a multibody articulated structure. Earth robots are of this type but they differ from spacecraft in that they usually have an Earth-ground reference. Slosh is the movement of a liquid in the spacecraft. Slosh can be due to propellant, heat pipes, or dampers that use a liquid to damp spacecraft angular rates.

Table 1.2 A one-semester course in attitude control systems.

Lecture	Chapter	Description
1	Preliminary Designs	Overview of ACS design. Requirements flow down. Interaction with other subsystems.
2	Control Theory	Introduction to control including loop shaping, Single-input–single-output, and state space.
3	Control Examples	Rate damper, PID, inner and outer loops.
4	Estimation theory background	Filtering, noise.
5	Estimation theory filters	Kalman filters linear, extended, and unscented.
6	Estimation theory examples	Kalman filter examples.
7	Orbit geometry and dynamics	Orbit geometry, orbit representation, orbit dynamics.
8	Orbit maneuvers	Maneuvers impulse, low thrust, close orbits.
9	Kinematics	Quaternions and transformation matrices, ephemeris.
10	Attitude dynamics introduction	Euler's equations.
11	Attitude dynamics advanced	Multibody, slosh, and flex.
12	Mechanisms	Reaction wheels, CMGs, deployment, robot arms.
13	Space environment	Atmosphere, radiation.
14	Budgets	Pointing budgets, disturbance budgets.
15	Actuators	Thrusters, reaction wheels, magnetic torquers, dampers.
16	Propulsion systems	RCS, delta-V, off-pulsing, blowdown.
17	Sensors	Sun sensors, Earth sensors, GPS, magnetometers.
18	Attitude control design	Overview of the process, simple 1-axis example.
19	Stellar attitude estimation	Camera processing, star-data processing centroiding, static solutions Kalman filters.
20	Simulation	Building simulations.
21	Testing	How ACS are tested. Spacecraft test flow.
22	Sun-nadir example	GPS IIR
23	Geosynchronous example	Momentum Bias.
24	Spinning satellite in transfer orbit example	Nutation-damping maneuvering.
25	Zero-momentum spacecraft example	3-axis, star cameras, Sun sensors, reaction wheels, thrusters.
26	Spacecraft operations	Examples of real-time remote operations.
27	Future work in ACS	Machine learning, advanced sensors, and actuators.

1.3. Courses based on this book

1.3.1 A one-semester course

The book fits into a one-semester course with twenty-seven 80-minute lectures. Table 1.2 gives a breakdown of lectures. The book includes MATLAB examples throughout and the students would receive the full source code for all of the scripts. Functions are p-files, that is precompiled functions. Supplementary readings would be technical papers and documents. The book has extensive references but no more than 10 papers would be assigned reading.

This material is used for an MIT course in attitude control. In the author's experience, it is best to focus on the basics in problem sets and projects. In this way, the students develop a base of knowledge that they can use.

1.3.2 A half-semester course

The course can be condensed into a one-semester course or a half-semester course with only 14 lectures. The number of examples is reduced and some lectures are combined. See Table 1.3.

Table 1.3 A half-semester course in attitude control systems.

Lecture	Chapter	Description
1	Preliminary Designs	Overview of ACS design. Requirements flow down. Interaction with other subsystems.
2	Control Theory	Introduction to control including loop shaping, Single-input–single-output, and state space.
3	Estimation theory background	Filtering, noise.
4	Estimation theory filters	Kalman filters linear, extended, and unscented.
5	Orbit geometry and dynamics	Orbit geometry, orbit representation, orbit dynamics.
6	Orbit maneuvers	Maneuvers impulse, low thrust, close orbits.
7	Kinematics	Quaternions and transformation matrices, ephemeris.
8	Attitude dynamics	Euler's equations.
9	Mechanisms	Reaction wheels, CMGs, deployment, robot arms.
10	Space environment	Atmosphere, radiation.
11	Actuators	Thrusters, reaction wheels, magnetic torquers, dampers.
12	Sensors	Sun sensors, Earth sensors, GPS, magnetometers.
13	Attitude control design	Overview of the process, simple 1-axis example.
14	Zero-momentum spacecraft example	3-axis, star cameras, Sun sensors, reaction wheels, thrusters.

1.3.3 An eight-lecture course

The course can be further condensed into an eight-lecture course suitable for an independent activity period or as a short summer course. This is the course the author teaches during MIT's Independent Activity Period. See Table 1.4.

Table 1.4 An eight-lecture course.

Lecture	Chapter	Description
1	Preliminary Designs	Overview of ACS design. Requirements flow down. Interaction with other subsystems.
2	Control Theory	Introduction to control including loop shaping, Single input single output, and state space.
3	Estimation theory filters	Kalman filters linear, extended, and unscented.
4	Orbit geometry and dynamics	Orbit geometry, orbit representation, orbit dynamics.
5	Attitude dynamics and Kinematics	Euler's equations, quaternions, external disturbances.
6	Space environment	Atmosphere, radiation.
7	Actuators and Sensors	Thrusters, reaction wheels, magnetic torquers, dampers, Sun sensors, star cameras, IMUs, GPS.
8	Zero-momentum spacecraft example	3-axis, star cameras, Sun sensors, reaction wheels, thrusters.

References

[1] ESA, Mercury Planetary Orbiter - spacecraft, https://sci.esa.int/web/bepicolombo/-/48872-spacecraft, March 2020.

[2] L. Szerdahelyi, S. Fugger, P. Espeillac, G. Monroig, T. Pareaud, M. Casasco, The BepiColombo attitude and orbit control system, in: 9th International ESA Conference on Guidance Navigation and Control Systems, 06 2014.

[3] ESA, Mercury Magnetospheric Orbiter at Mercury, https://sci.esa.int/web/bepicolombo/-/60079-mercury-magnetospheric-orbiter-at-mercury, September 2019.

CHAPTER 2

History

2.1. Space story

My father took my brother and me to a showing of Stanley Kubrick's movie, "2001: A Space Odyssey" at the Cinerama Theater in New York City. The movie was filmed in Super Panavision 70 with some scenes in Todd-AO and MCS-70. The Cinerama theater showed the film on a curved screen helping to immerse the audience in the film. The theater itself was one of the first "movie palaces" that looked like a theater where a ballet or opera might be performed but were designed only for movies. The first 20 minutes were a surprise as were the last 20 minutes or so. In between, the movie was an excellent tutorial on the future of spaceflight.

2.2. Introduction

It is hard to be a spacecraft engineer unless you know the history of spacecraft engineering. Knowing history allows you to learn from past successes and mistakes. It also helps prevent you from "reinventing the wheel" instead of taking what has been done and building on that with your innovations. It is also a lot of fun to see what engineers and scientists did when they were first learning the elements of spacecraft design. Many of the engineers went from designing aircraft to spacecraft. While spacecraft and aircraft are different, the elements of design and the methodologies of design do not change. The cancelation of the Canadian supersonic C-105 Arrow fighter program resulted in several very talented Canadian engineers moving to the United States to work on Apollo. This chapter will include comments on the control aspects of the designs.

We will start with ancient history. Some will be in the realm of fiction with the first ideas of how one might travel in space and what adventures might be found. The remaining sections will cover the "Space Race" and the beginnings of automated spacecraft and robotic exploration. Engineers were learning how to make spacecraft autonomous back in the 1950s and 1960s without the use of digital computers. Digital computers enabled the Apollo missions. The chapter will continue to the present day and give some ideas on where the space business will go in the future.

2.3. Pre-1950 – dreaming of space

Robert Goddard pioneered rockets, closely followed by the Germans led by Werner von Braun. The first applications were military, the V-2 for example. However, all of

https://doi.org/10.1016/B978-0-32-399915-1.00012-7

the pioneers were interested in space. Sängers' Antipodal bomber, a boost-glide vehicle designed to bomb the United States, was also a potential launch vehicle. It was the inspiration for the German Sänger II launch vehicle. Sänger II was to be fully reusable with a hypersonic turboramjet-powered first stage and an LH_2/LO_2 second stage. It would have flown from Europe to the equator at Mach 4 and then accelerated to Mach 6.7 before releasing the second stage. It was canceled because it did not appear to offer any cost advantages over the Ariane 5.

A multistage V-2 might have reached orbit, and later effectively did as part of both the US and Soviet space programs. The first successful launch vehicles had analog control systems and very few ways to test them, aside from actually flying the vehicle. The number of failures should be thought of in terms of how many bugs there are in the average modern software package.

2.4. 1950s – getting started

Intercontinental Ballistic Missiles (ICBMs) drove the space programs of both the United States and the Soviets. While bombers could deliver nuclear weapons, they were vulnerable to ground-to-air missiles and fighters. ICBMs gave an opponent only minutes to make a decision and shooting one down remains an elusive goal to this day. Soviet bombs were bigger than US bombs so they built bigger boosters, which gave them an edge in the early days of the space race. Sputnik was launched, alarming the United States and leading to the Space Race.

2.5. 1960s – the Golden Age: Apollo to the moon

Vostok 1 kicked off the space race big time, even more than Sputnik. Space was in the news and on TV in the 1960s. It was so important that engineers and scientists involved in the programs were celebrities. Hence the term "Rocket Scientist".

Both the United States and Soviet Union launched crewed and robotic missions. Those with crews included the Vostok, Soyuz, Mercury, Gemini, and Apollo spacecraft. The robotic missions included the first landings on the moon and missions to Venus and Mars. Oddly, Venus has been a very successful Russian destination, while Mars has been pretty much a US destination.

Commercial activities began with the first commercial communications satellites, which remain a big and profitable business. It may still be the only profitable space business. Three American companies, Ford Aerospace, RCA Astro-Electronics, and Hughes Space and Communications were the leaders in the communication-satellite business in the 1960s but the Europeans and Japanese followed quickly with their communication satellites. Now, communication-satellite operators have a huge number of choices for their communication satellites.

The US Apollo moon program was a driving force in space in the 1960s. One of the first issues in human spaceflight was how much control, if any, to give the pilot. The engineers were quite confident that their automatic control systems could work successfully without human intervention. They were concerned the pilot would make any situation worse. This belief was a fundamental error based on a misunderstanding of control. You only need closed control to deal with uncertainties in the system. Pilots are adaptable. Their control systems were not. At least in the US program, the crews were essential.

Apollo had the first fully digital control system. The Charles Stark Draper Laboratory had to build its computer from medium-scale integrated circuits such as a NAND gate on a chip. Memory was magnetic cores woven by textile workers. Draper also designed the programming languages. The flight computer could not be reprogrammed in flight, leading to numerous workarounds. One of the most famous was the stuck abort switch on Apollo 14. In a very short amount of time, MIT software engineers came up with a workaround to convince the flight computer that the landing had already been aborted, thus preventing from aborting the landing, which went on as scheduled. Don Eyles, the lead on the team that solved the problem, was awarded a medal by the City of Boston for his quick and innovative work. Both the Apollo Command Module and Lunar Module used phase-plane controllers. These were easy to implement using digital computers. Example 2.1 shows a phase-plane controller that is similar to the Lunar Module system. The Lunar Module had a clever system, developed by Richard Goss [1], that allowed a 3-axis phase-plane controller to work effectively given the Lunar Module thruster geometry.

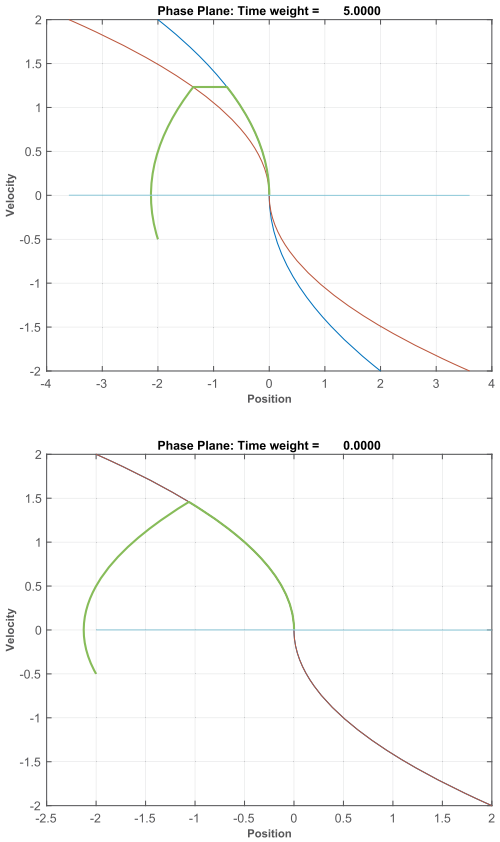
An important aspect was manual control of the Lunar Module. Robert Stengel [2] led its design. Humans in the loop are equally as important on remote spacecraft, though typically a lot less thought is given to the human/spacecraft interface for robotic spacecraft.

It is easy to forget that before Apollo, controls engineers thought in terms of block diagrams that could be turned into analog control systems using operational amplifiers. This led to a long list of block diagram-oriented products like Control-C, Matrix-X, and Simulink. Unfortunately, much like you can write spaghetti code, you can write spaghetti block diagrams. As flight-control systems start using expert systems and neural networks, block diagrams will be even less useful in the future.

NASA had ambitious plans for the moon including a nuclear thermal upper stage for the Saturn V. Early versions of nuclear thermal stages were ground tested. Robotic versions of the Lunar Module were planned for landing large masses of cargo. Unfortunately, the Apollo program was canceled after Apollo 17 due to the perception that it had lost public support.

The Defense Meteorological Satellite Program (DMSP) began in the 1960s at RCA Astro-Electronics in NJ. The satellites went through many changes over their long

```
n               = 8000;
xP              = zeros(3,n);
dT              = 0.001;
x               = [-2;-0.5];
dRHS            = struct('inr',3,'u',0);
kOpt            = 0;

for k = 1:n
  u = PhasePlane(x(2),x(1),kOpt,0.001,0.001);
  xP(:,k) = [x;u];
  dRHS.u = u;
  x = RK4(@RHS,x,dT,0,dRHS);
end

PhasePlanePlot( xP(2,:), xP(1,:),kOpt
      ,0.001,0.001,[-2,2])


%% Right hand side [ang1;ang2;omega1;omega2]
function xDot = RHS(x,~,d)

xDot    = [x(2);d.u];

end
```



Example 2.1: An Apollo Lunar Module-type phase-plane controller. The red and blue lines are the optimal switching lines. They are separated in the first plot because of the fuel consideration of the optimal control. The second plot shows the time optimal switching curves.

history. The TIROS satellite was the NASA version. Eventually, they merged into a common spacecraft. The NASA version used an Earth sensor, while the Air Force system had a star camera and an IMU-based Kalman filter for attitude determination. The star camera was a v-slit design and identified stars using the star transits of the slits. It had an 80-star catalog.

2.6. 1970s – the Space Shuttle era

Skylab and the Apollo-Soyuz missions were the immediate follow-ons to Apollo using leftover hardware. The Space Shuttle was the follow-on to Apollo. NASA had its eyes on lunar bases and human missions to Mars but ended up with the Shuttle that was sup-

posed to lower the cost of getting payloads and humans into space, thus enabling lunar and Mars missions. The Shuttle, despite two tragic failures, was a successful program. The Shuttle launched Hubble and many modules for the International Space Station (ISS). Another program, GPS, was launched in 1973. Arguably it has had the greatest impact of any space program aside from communications satellites. The first US space station, Skylab, was launched and had three different crews. The Soviets launched their Molniya space station. Many Shuttle-centric programs, including a reusable upper stage, were started and then canceled after the Challenger explosion. The Air Force, which was the reason for the size of the Shuttle and its wing shape, baled out of the Shuttle program as they never really wanted to use it under any circumstances.

The Shuttle was developed at the beginning of the computer era. Simulations for the Shuttle Orbiter ran on mainframes and were limited to about 90 seconds of simulated time. The Shuttle Landing simulator at Draper Laboratory was an analog computer simulation. One of the Draper software engineers said, "The only thing you can be sure about with an analog computer is that today's results won't match yesterday's results."

The Shuttle introduced the HAL/S (High-order Assembly Language/Shuttle) programming language [3]. The official story is that it is named after Hal Laning of Draper Lab but by coincidence, HAL/9000 of 2001 fame was a notable Artificial-Intelligence movie character of the time. It was a high-level language with innovations such as explicit type definition, long variable names, three lines of print per line of code (for exponents and subscripts), and other interesting features.

```
E          2         2
M    X = A    +   B
S                   I
```

The equivalent MATLAB® code would be

```
x = a^2 + b(i)^2;
```

The Space Shuttle also used an IBM 360-based flight computer. It had limited storage and every change of mission phase required a new load of software into the flight computer.

HAL/S was a competitor for the Air Force standard language. Ada (named after Ada Lovelace the first programmer), based on Pascal, ended up winning. HAL/S was used on the NASA Galileo spacecraft [4] but that was the last major application. C, a Bell Labs creation, and later C++ and object-oriented version, eventually won the race for standard languages.

From a control point of view, the Shuttle has considerable heritage from the Apollo GN&C system. It continued to use the phase-plane control system invented for Apollo. The addition of the arm required that the phase-plane update its angular increments (the amount of attitude angle changed per pulse) every time the arm moved since the inertia changes as the arm moves.

The Space Shuttle itself was an international program. The Canadian Space Agency supplied the robot arm, called Canadarm, which was an incredible achievement in robotics. It also taxed the Shuttle's on-orbit autopilot that was not designed with a huge, flexible structure in mind that also changed the Orbiter inertia matrix whenever it moved.

The Voyager spacecraft were launched and are still in operation today. They entered interstellar space and, for the moment, are the first human-built objects that any interstellar space-faring races will encounter, though the NASA New Horizons mission is catching up!

2.7. 1980s – internationalization

The Soviet/American space race was the focus of the 1960s and into the 1970s. Arianespace was founded in 1980 to compete with the US, Soviet, and Chinese launch industries. It was the first major European space program. Since then the European presence in space has grown dramatically. At the same time, Japan's and India's programs grew. Anything the United States could do, they could do. China began working on its program and despite the best efforts for "export control" launched a space station and the first lunar far-side lander. India has been particularly adept at developing low-cost missions. China has also flown its space station.

The 1980s saw competition between RCA, Hughes, and Ford for the communication-satellite market. Hughes had a dual-spin spacecraft, the HS 376, to compete with RCA's momentum-bias Series 2000 through 5000 designs. All three companies supplied most of the world's comsats. All three used spinning parts to give their spacecraft gyroscopic stiffness. Bias momentum is very handy when your spacecraft loses Earth lock because it will not tumble into a random orientation without the Sun on the solar panels.

The Iridium constellation was conceived in 1987 and launched in 1998. It was 66 satellites reduced from the original 77, hence the name. It flew just in time to compete with cell towers so it became a financial failure. It was saved because the US military was using it for communications for forces in remote locations. Its latest version tracks commercial aircraft around the globe along with its communication payload.

2.8. 1990s – Hubble

The Hubble Space Telescope was based on US Air Force telescopes except that it pointed away from the Earth. It was designed to fly on the Space Shuttle mostly because the Space Shuttle was designed to carry similar-sized reconnaissance satellites. Hubble was designed to be maintained by astronauts. This was a good thing because it launched with astigmatism. It was repaired by astronauts several times and continues to operate

to this day. The Hubble used reaction wheels for control. The reaction wheels, which use brushless DC motors, were so noisy that they had to be put on suspensions. Fig. 2.1 shows Hubble. The James Webb Space Telescope uses a similar system.

Figure 2.1 Hubble Space Telescope. Image courtesy of NASA.

Gravity Probe B, a Stanford satellite, was built by Lockheed Martin to test Einstein's General Theory of Relativity. It had a cryogenic Inertial Measurement Unit (IMU) and was so noise sensitive that reaction wheels were ruled out. Instead, it used the helium boil-off from the Inertial Measurement Unit (IMU) dewar for control. The mission was a spectacular success and demonstrated, once again, that General Relativity is correct.

In 1992 Mars Observer was launched. It used elements of the DMSP control system and some features of RCA/GE communications satellites. It was the first real attempt to commercialize space-science missions, that is to have an outside vendor design and build an interplanetary spacecraft with minimal government oversight. Mars Observer was the first interplanetary satellite with reaction wheels. It failed on Mars entry when the propulsion system exploded. The attitude control system needed a last-minute fix to deal with slosh in the fuel tanks. A full-state feedback controller was replaced with a simpler PID controller with roll-off filters to prevent slosh interaction.

1998 was the year of the launch of the central module of the International Space Station (ISS), shown in Fig. 2.3. This would grow into the largest human-built object in space. It also demonstrated even more than Hubble, that humans could work in free fall. The ISS uses double-gimbaled control-moment gyros (CMGs) for control. It also has a reaction control system, which is a satellite that uses thrusters. CMGs require complex

steering logic to keep them out of gimbal lock. Single-gimbal CMGs are often used on Earth-pointing spacecraft in LEO that need to track a target on the ground. Fig. 2.2 shows the ISS CMGs.

Figure 2.2 Four ISS Control-Moment Gyros. Image courtesy of NASA.

You might also observe that the ISS cannot be modeled as a rigid body with all the flexible appendages such as the solar wings and the radiators. Once you add the robots crawling up and down the trusses, the robot arms, etc., it becomes a control engineer's nightmare.

Fig. 2.4 shows the Japanese H-II transfer vehicle. It docks to the ISS via a robot arm.

1999 was the year of the CubeSat Program [5]. CubeSat standard was created by California Polytechnic State University and Stanford University's Space Development Laboratory. Professors Jordi Puig-Suari of Cal Poly and Bob Twiggs of Stanford also realized that small satellites made excellent student projects, and could be useful in their own right. Publishing of a standard simplified the design of CubeSats. The first CubeSats were launched in 2003. This spawned a large and growing industry. One important aspect was that CubeSats greatly lowered the cost of entry to space, leading to innovations in every aspect of space flight.

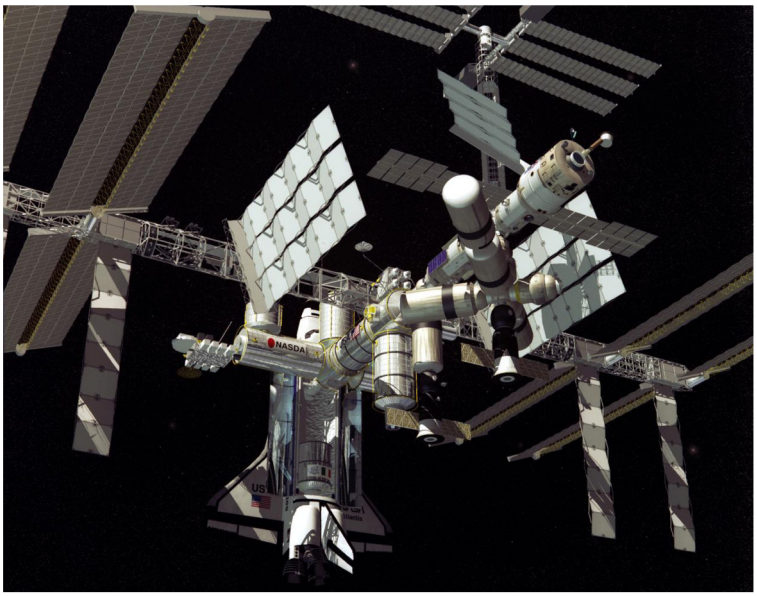


Figure 2.3 The International Space Station. Image courtesy of NASA.

2.9. 2000s – commercial space is reborn

CubeSats started booming with the benefit of helping to train a new generation of space engineers. People started companies for commercial space imaging, commercial science, and other commercial ideas. Space X was founded in 2002. After three launch failures, Falcon 1 successfully reached orbit. This made it feasible for NASA to award a contract to Space X to build the Falcon 9 that has become a space-launch workhorse. While arguably a government-funded entity, Space X works essentially on its own without dozens of NASA design reviews.

The ISS continued to be built, becoming the largest space project ever. Modules from many countries were integrated into the ISS. Japan supplied the JEM module. The ISS was resupplied with the Space Shuttle, Russian Progress spacecraft, the Japanese H-II Kountori Transfer Vehicle, the Dragon resupply vehicle, and the ESA ATV.

The United States continued its exploration of Mars with landers and orbiters. JPL's rovers have covered wide swaths of territory and proven to be reliable explorers. New Horizons, the Pluto explorer, was launched in 2009.

2.10. 2010s – the space station and beyond

The United States continued its exploration of Mars. New Horizons flew by Pluto on July 14, 2015, and continues to explore the edge of the solar system. The very am-

Figure 2.4 Japanese H-II transfer vehicle. The use of space robot arms was pioneered by NASA and the Canadian Space Agency for the Shuttle. Image courtesy of NASA.

bitious James Webb Space Telescope was started and launched in 2021. The United States looked into a variety of follow-ons to the Shuttle. The Orion space capsule and Space Launch System was started. At the same time, the number of commercial entities involved in space grew.

2.11. The future

Five modules are still to be launched to ISS. The United States has funded commercial crewed and robotic lunar landers. There is talk of human Mars missions. China plans to send crews to the Moon and Mars. Interest in interstellar missions, probably robotic, has grown with some NASA money going into preliminary studies. The limitation to more ambitious missions is propulsion, which is difficult. Some money is being spent on nuclear electric and fusion propulsion systems that have the potential to revolutionize space exploration. Fig. 2.5 is a NASA-funded design for a Pluto orbiter powered by a fusion rocket engine. SpaceX wants to send people to Mars by 2029. NASA hopes to land people on the Moon, again, around the same time.

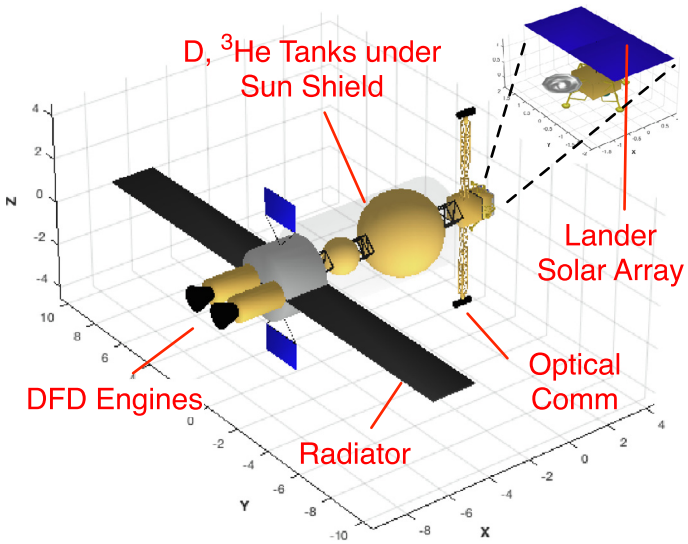


Figure 2.5 Pluto orbiter. DFD stands for Direct Fusion Drive.

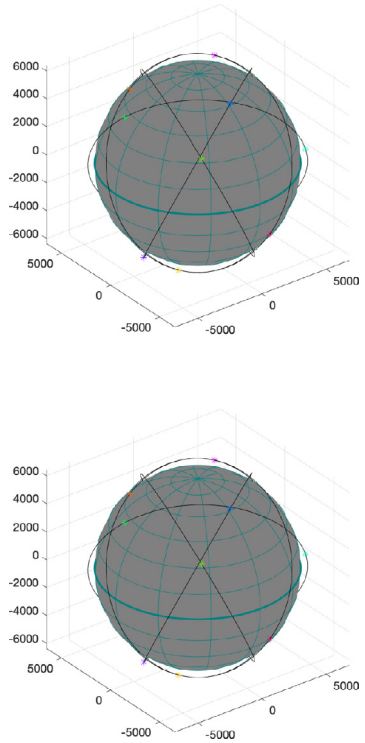
The United States is funding three commercial companies to build human lunar landers. NASA is also funding SpaceX to build commercial lunar landers.

A large area of interest is in Low-Earth Orbit constellations. These involve hundreds, if not thousands, of satellites. For communications, the satellites must connect to each other, to ground stations, and user terminals on the ground. Example 2.2 shows two types of constellations. SpaceX is flying an enormous constellation for the Internet called Starlink. Amazon plans a similar constellation.

Space exploration has expanded beyond its beginnings as US/Soviet competition. It is now a truly international effort. China landed a lander with a rover on the far side of the moon and has launched communications, navigation, and scientific satellites into Earth orbit. The United Arab Emirates sent a mission to orbit Mars. India has also launched Mars missions and built a full range of satellites for every application. The European Space Agency (ESA) has built many advanced satellites, including lunar orbiters, communication satellites, and navigation satellites. The Swedish Space Corporation flew an ambitious rendezvous robots experiment and the first electric-propeled lunar mission. The ESA is building the service module for the Orion spacecraft for the Artemis lunar missions.

From a controls perspective, spacecraft have gone from passive control to analog control to digital control. Autonomy has always been a spacecraft requirement but even today many spacecraft require extensive interaction with the ground for operation. As the missions get more complex there will need to be more sophisticated onboard systems

```
%% Walker Constellation

WalkerConstellation
```



Example 2.2: Walker and polar-star constellations.

for control, fault detection, and mission planning. This book will help you build your background to address future challenges.

References

[1] J.M. Dahlen, P.G. Felleman, R.D. Goss, N.E. Sears, M.B. Trageser, Guidance and Navigation System for Lunar Excursion Module, Tech. Rep. R-373, MIT Instrumentation Laboratory, July 1962.
[2] R.F. Stengel, Manual attitude control of the lunar module, Journal of Spacecraft and Rockets 7 (8) (1970) 941–948.
[3] M.J. Ryer, Programming in HAL/S, Tech. Rep. NAS9-13864, NASA Jet Propulsion Laboratory, September 1978.
[4] Computers in Spaceflight: The NASA Experience, https://www.history.nasa.gov/computers/contents.html, 2009.
[5] The CubeSat Program, https://www.cubesat.org/about.

CHAPTER 3

Single-axis control

3.1. Space story

We started up a station-keeping maneuver on a Series 5000 spacecraft for the first time. Series 5000 were the latest GE momentum-bias spacecraft. Once the maneuver started, the performance was not what we expected so we shut it down. This was the first GE communications satellite with a Proportional-Integral-Derivative Controller. What we observed was due to a PID being conditionally stable and the minimum impulse being higher than expected. As it turned out, when you opened the valve just a little it burned more fuel getting more impulse per pulse. This was easy to fix. What was interesting was that the time response looked like a single-axis double integrator. This is the motivation for this chapter.

3.2. Introduction

This chapter presents a complete design of a single-axis control system including a Kalman filter for attitude determination. The goal is to present a simple system that illustrates many aspects of spacecraft control. It gives you code that you can explore to improve your understanding of spacecraft control. The overall system is shown in Fig. 3.1.

This controller is configured as a regulator. It maintains the angle but does not have a set point input. The set point is always zero.

3.3. Dynamical systems

The two dynamical systems that need to be modeled in this example are the spacecraft and the gyro. The spacecraft model is a double integrator

$$I\ddot{\theta} = T_c + T_d \tag{3.1}$$

or

$$\begin{aligned} \dot{\theta} &= \omega \\ I\dot{\omega} &= T_c + T_d \end{aligned} \tag{3.2}$$

where θ is the true angle and ω is the true angular rate, I is the inertia, T_c is the control torque, and T_d is the disturbance torque.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00013-9

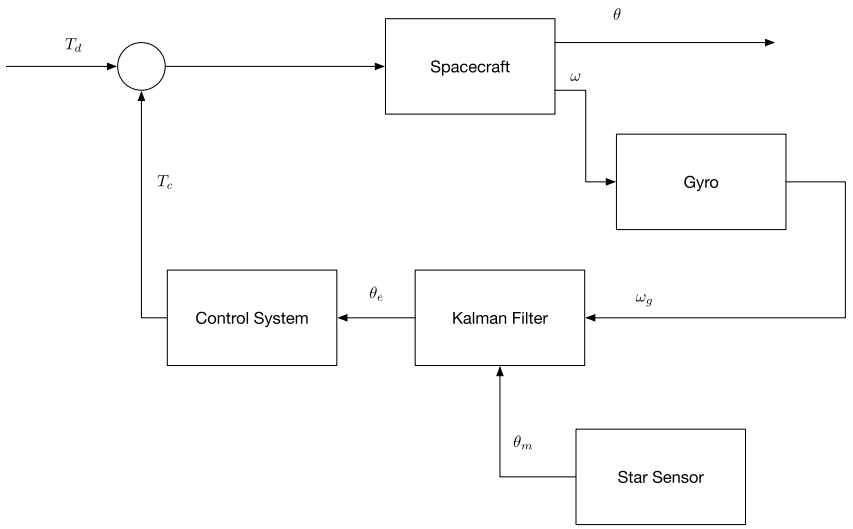


Figure 3.1 Single-axis control system.

The gyro model is a single integrator

$$\begin{aligned}\dot{\theta}_g &= \omega + b + \nu_\theta \\ \dot{b} &= \nu_b\end{aligned} \tag{3.3}$$

where θ_g is the integral of the true angular rate, ω plus the bias b. The bias is driven by white noise, meaning that it is a random-walk process. ν_b is the bias noise input. ν_θ is the angle white noise.

3.4. Control system

The control system is a proportional-derivative control system. In discrete time the controller is

$$T_c[k] = -K_f\left(\theta_e[k] + \tau_r\left(\frac{\theta_e[k] - \theta_e[k-1]}{\tau}\right)\right) \tag{3.4}$$

where K_f is the forward gain, τ_r is the rate damping time constant, τ is the controller time step, and k indicates the sample. If we write the continuous closed-loop system in the s-plane we get the second-order equation

$$Is^2 + K_f\tau_r s + K_f = 0 \tag{3.5}$$

We can equate the gains with the undamped natural frequency and damping ratio

$$K_f = \omega_n^2 I \tag{3.6}$$

$$\tau_r = \frac{2\zeta\omega_n I}{K_f} \tag{3.7}$$

where ω_n is the desired undamped natural frequency and ζ is the damping ratio. If the sampling rate is sufficiently fast, these gains will give the desired response and we will not have to consider sampling time. Sufficiently fast is a sampling frequency much faster, that is five to ten times faster, than the control bandwidth, ω_n.

3.5. Kalman filter

The Kalman filter has two parts, state prediction and measurement incorporation. The gyro model in state-space form is

$$x_k = ax_{k-1} + b\omega_{k-1} \tag{3.8}$$

$$y = hx \tag{3.9}$$

where x is the Kalman filter state vector, h is the measurement matrix, and a is the state-transition matrix. It implements one time step for the state transitioning it from the state at $k-1$ to that at k. Expanding the matrices

$$\begin{bmatrix} \theta_g \\ b \end{bmatrix}_k = \begin{bmatrix} 0 & \Delta t \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_g \\ b \end{bmatrix}_{k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} \omega_{k-1} \tag{3.10}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta_g \\ b \end{bmatrix} \tag{3.11}$$

The state vector consists of the angle θ_g and the bias b. The equations show that the true angular rate and the bias both cause the state to change.

The Kalman-filter measurement incorporation step is

$$k = \frac{ph^T}{hph^T + r} \tag{3.12}$$

$$x_e = k(y - hx_e) \tag{3.13}$$

$$p = (E - kh)p \tag{3.14}$$

where E is a 2-by-2 identity matrix, p is the covariance matrix, r is the measurement covariance matrix, and the measurement matrix is h. The prediction step is

$$x_e[k] = ax_e[k] + b\left(\frac{\theta_i[k] - \theta_i[k-1]}{\Delta t}\right) \tag{3.15}$$

$$p = apa^T + q \tag{3.16}$$

where a is the state-transition matrix, q is the model-covariance matrix, and Δt is the time step. The model-covariance matrix is a Gaussian white-noise approximation to all

of the uncertainty in the model, including unmodeled inputs. The output of the rate-integrating gyro is differenced to get the rate. This is where the IMU connects to the model. The IMU output is not a measurement.

An important point is that the covariance follows a path predetermined by the measurement covariance, the model covariance, and the initial covariance. Essentially, it is a variable-gain filter that will reach steady-state values and stay there forever.

3.6. Simulation

Example 3.1 gives a simulation with the Kalman filter in the loop. The time step is chosen based on the bandwidth of the controller. It must be

$$\Delta T < \frac{1}{2}\frac{2\pi}{\omega_n} \tag{3.17}$$

This is known as the Nyquist sampling frequency. This can be understood by looking at a sine wave. Fig. 3.2 shows a sine wave. If we sampled at half the period, there is a possibility that we would see only zeros. Thus, we have to sample more frequently.

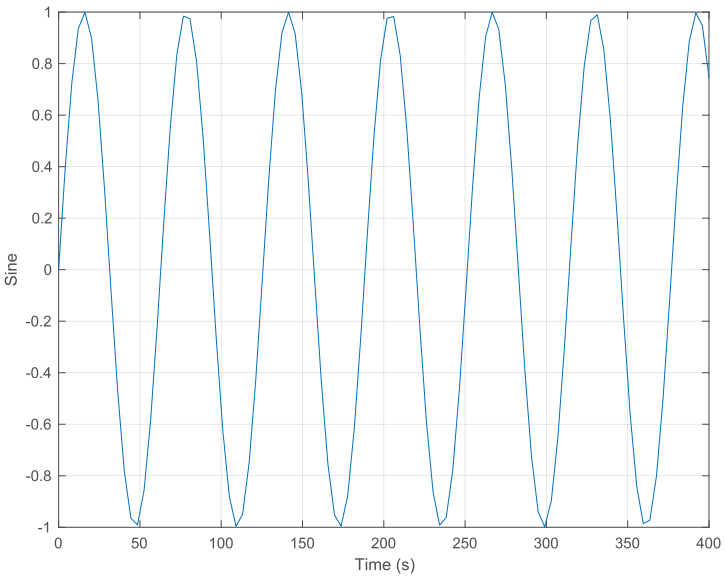
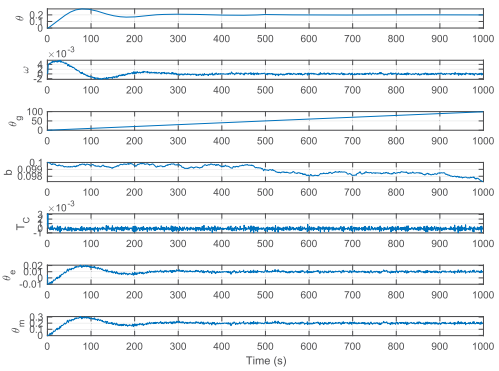


Figure 3.2 Sine wave to illustrate sampling. If sampling was exactly half the period, the samples might all be zero.

A step disturbance is applied to the spacecraft. The gyro measures the angular rate but its output includes the gyro bias, which is modeled with a random walk, and the angle noise. The integrated rate is used by the control system. Note that the integrated angle, from the gyro, grows with time due to the bias.

```
%% Simulation of a single axis spacecraft

%% Dynamical model data
tD    = 1e-4; % Disturbance torque (Nm)
dRHS = struct('inertia',1,'tC',0,'tD',tD,'
       sigTheta',0.00001,'sigB',0.0001);
sigM = 0.01; % Measurement noise

%% Design the controller
zeta = 1.0;
wN   = 0.1;
kF   = wN^2*dRHS.inertia;
tauR = 2*zeta*wN*dRHS.inertia/kF;
thetaOld = 0;

% Kalman Filter
dT   = 1;
p    = diag([0.001 0.0001]); % Initial covariance
q    = 0.01*p; % Plant covariance
dKF = struct('p',p,'q',q,'r',sigM^2,...
   'thetaE',0, 'xE',[0;0],'a',[0 dT;0 0],...
   'b',[dT;0], 'h',[1 0],...
   'thetaGOld',0,'dT',dT);

% Time step
n    = 1000;
xP   = zeros(7,n);

%% Simulation
x  = [0;0;0;0.1];% [angle;rate;gyro angle;bias]
for k = 1:n
   thetaM   = x(1) + sigM*randn; % Measurement
   thetaG   = x(3); % Gyro angle
   dKF      = KalmanFilter(thetaG,thetaM,dKF);
   dRHS.tC  = -kF*(dKF.thetaE + tauR*(dKF.thetaE-
              thetaOld)/dT);
   thetaOld = dKF.thetaE;
   xP(:,k)  = [x;dRHS.tC;dKF.thetaE;thetaM];
   x        = RK4(@RHS,x,dT,0,dRHS);
end

%% Plotting
t  = (0:n-1)*dT;
yL = {'\theta','\omega','\theta_g','b','T_C',...
   '\theta_e','\theta_m'};
TimeHistory(t,xP,yL,'SingleAxisSimulation')

%% Right hand side
function xDot = RHS(x,~,d)
omega    = x(2);
b        = x(4);
xDot     = [omega;(d.tC+d.tD)/d.inertia;...
            omega+b+d.sigTheta*randn;...
            d.sigB*randn];
end

%% Kalman Filter
function d = KalmanFilter(thetaG,y,d)
% State propagation
omega       = (thetaG - d.thetaGOld)/d.dT;
d.thetaGOld = thetaG;
d.xE        = d.a*d.xE + d.b*omega;
d.p         = d.a*d.p*d.a' + d.q;
% Measurement incorporation
k           = d.p*d.h'/(d.h*d.p*d.h'+d.r);
d.xE        = k*(y - d.h*d.xE);
d.p         = (eye(2) - k*d.h)*d.p;
d.thetaE    = d.xE(1);
end
```



Example 3.1: Single-axis step response.

3.7. Adding a mode

We can add a second wheel to the dynamical model. This could represent slosh or a mode due to structural flexibility. One way to derive the equations of motion of the new system is through angular-momentum conservation. Write the total angular momentum of the total spacecraft and that of just the second wheel.

$$H = I\dot{\theta} + J(\dot{\psi} + \dot{\theta}) \tag{3.18}$$

$$H_j = J(\dot{\psi} + \dot{\theta}) \tag{3.19}$$

where J is the inertia of the second wheel. The equations of motion are found by taking the derivatives of H and H_j.

$$T_d + T_c = I\ddot{\theta} + J(\ddot{\psi} + \ddot{\theta}) \tag{3.20}$$

$$T_j = J(\ddot{\psi} + \ddot{\theta}) \tag{3.21}$$

where T_j is the internal torque on the second wheel and ψ is the angle of the second wheel with respect to the core body. These can be written as

$$T_d + T_c = I\ddot{\theta} + T_j \tag{3.22}$$

$$T_j = J(\ddot{\psi} + \ddot{\theta}) \tag{3.23}$$

or

$$\begin{bmatrix} I+J & J \\ J & J \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} T_c + T_d \\ T_j \end{bmatrix} \tag{3.24}$$

The matrix on the left side is known as the generalized inertia matrix. The generalized inertia matrix is symmetric. Let T_j be a general spring and damper torque,

$$T_j = -c\dot{\psi} - k\psi \tag{3.25}$$

We can then write out the state equations and get the eigenvalues. The eigenvalues are the natural frequencies of the system.

$$\begin{bmatrix} \dot{\theta} \\ \dot{\psi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{k}{I} & 0 & \frac{c}{I} \\ 0 & -\frac{k}{\gamma} & 0 & -\frac{c}{\gamma} \end{bmatrix} \begin{bmatrix} \theta \\ \psi \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.26}$$

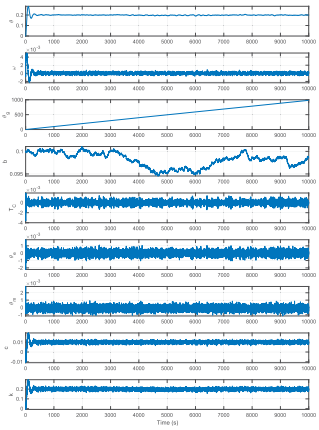
where

$$\gamma = \frac{IJ}{I+J} \tag{3.27}$$

If $I >> J$, $\gamma = J$. The script Example 3.2 shows the response with the added model. We compute the eigenvalues to determine the model frequency. The parameters, in this

```
%% Simulation of a single axis spacecraft
% Includes a wheel

%% Dynamical model data
tD        = 1e-4; % Disturbance torque (Nm)
wJ        = 0.6;
zetaJ     = 0.1;
inertiaJ  = 0.001;
k         = inertiaJ*wJ^2;
c         = 2*zetaJ*wJ*inertiaJ;
dRHS = struct('inertia',1,'tC',0,'tD',tD,'
       sigTheta',0.00001,...
       'sigB',0.0001,'inertiaJ',inertiaJ ,'k',k,'
            c',c);
sigM = 0.01; % Measurement noise

%% Design the controller
zeta = 1.0;
wN   = 0.1;
kF   = wN^2*dRHS.inertia;
tauR = 2*zeta*wN*dRHS.inertia/kF;
thetaOld = 0;

%% Eigenvalues
gamma = dRHS.inertiaJ*dRHS.inertia/(dRHS.inertiaJ
        + dRHS.inertia);
a     = [0 0 1 0;0 0 0 1;...
  0 dRHS.k/dRHS.inertia 0 dRHS.c/dRHS.inertia;...
  0 -dRHS.k/gamma          0 -dRHS.c/gamma];
eig(a)

% Kalman Filter
dT   = 1;
p    = diag([0.001 0.0001]); % Initial covariance
q    = 0.01*p; % Plant covariance
dKF = struct('p',p,'q',q,'r',sigM^2,...
  'thetaE',0, 'xE',[0;0],'a',[0 dT;0 0],...
  'b',[dT;0], 'h',[1 0],...
  'thetaGOld',0,'dT',dT);

% Time step
n    = 10000;
xP   = zeros(9,n);

%% Simulation
x  = [0;0;0;0.1;0;0];% [angle;rate;gyro angle;
      bias]
for k = 1:n
  thetaM   = x(1) + sigM*randn; % Measurement
  thetaG   = x(3); % Gyro angle
  dKF      = KalmanFilter(thetaG,thetaM,dKF);
  dRHS.tC  = -kF*(dKF.thetaE + tauR*(dKF.thetaE-
         thetaOld)/dT);
  thetaOld = dKF.thetaE;
  xP(:,k)  = [x;dRHS.tC;dKF.thetaE;thetaM];
  x        = RK4(@RHS,x,dT,0,dRHS);
end

%% Plotting
t  = (0:n-1)*dT;
yL = {'\theta','\omega','\theta_g','b','T_C',...
  '\theta_e','\theta_m','c','k'};
TimeHistory(t,xP,yL,'SingleAxisSimulation')

%% Right hand side
function xDot = RHS(x,~,d)
omega    = x(2);
b        = x(4);
psi      = x(5);
psiDot   = x(6);

tJ       = -d.k*psi - d.c*psiDot;
omegaDot = (d.tC+d.tD - tJ)/d.inertia;
xDot     = [omega;omegaDot;...
            omega+b+d.sigTheta*randn;...
            d.sigB*randn;...
            psiDot;...
            tJ/d.inertiaJ - omegaDot];
end

%% Kalman Filter
function d = KalmanFilter(thetaG,y,d)
% State propagation
omega       = (thetaG - d.thetaGOld)/d.dT;
d.thetaGOld = thetaG;
d.xE        = d.a*d.xE + d.b*omega;
d.p         = d.a*d.p*d.a' + d.q;
% Measurement incorporation
k           = d.p*d.h'/(d.h*d.p*d.h'+d.r);
d.xE        = k*(y - d.h*d.xE);
d.p         = (eye(2) - k*d.h)*d.p;
d.thetaE    = d.xE(1);
end
```

```
>> SingleAxisWithWheelSim

ans =

   0.0000e+00 + 0.0000e+00i
   0.0000e+00 + 0.0000e+00i
  -6.0060e-02 + 5.9729e-01i
  -6.0060e-02 - 5.9729e-01i
```



Example 3.2: Single-axis step response when there is a second mode. The eigenvalues are given. It has one oscillatory mode and one rigid-body mode.

example, do not represent any particular spacecraft. The undamped natural frequency of the mode is six times the controller bandwidth.

A step disturbance is applied to the spacecraft. The mode is excited but damps out quickly. The presence of the mode leads to a noisier response.

It is useful to look at the transfer function between torque and angle. Write

$$T = (I+J)s^2\theta + Js^2\psi \tag{3.28}$$

$$0 = J(s^2\psi + s^2\theta) + cs\psi + k\psi \tag{3.29}$$

where all angles are functions of s. Solving for ψ using the second equation

$$\psi = -\frac{Js^2\theta}{Js^2 + cs + k} \tag{3.30}$$

we get

$$T = (I+J)s^2\theta - \frac{Js^4\theta}{Js^2 + cs + k} \tag{3.31}$$

or

$$T = \left(\frac{(I+J)s^2(Js^2 + cs + k) - J^2s^4}{Js^2 + cs + k}\right)\theta \tag{3.32}$$

The angle response is therefore

$$\frac{\theta}{T} = \left(\frac{1}{(I+J)s^2}\right)\left(\frac{Js^2 + cs + k}{\gamma s^2 + cs + k}\right) \tag{3.33}$$

The first part is the rigid-body response, which is a double integrator. The second part is a pole zero pair. If $I >> J$ this is 1 since the zero will be very close to the pole. The zero will then cancel the pole, making it invisible to the control system. This analysis could be expanded to any number of modes. This particular example shows that the existence of a flex or slosh mode does not necessarily impact the closed-loop response. However, in every case, the influence needs to be determined through analysis.

CHAPTER 4

ACS system design

4.1. Introduction

This chapter describes the process of creating a preliminary control-system design. It describes how to interpret customer requirements, how to select actuators and sensors, and how to organize your design. Designing an attitude control system from scratch presents the control-systems engineer with a bewildering set of choices. The engineer must select:

- Spacecraft type (dual-spin, momentum-bias, zero-momentum, Sun-nadir, etc.);
- Control strategy (active or passive);
- Actuators (wheels, magnetic torquers, thrusters, etc.);
- Sensors (Earth, star, Sun, planet, magnetic field, gyros etc.);
- Processors (1750A, 80386, 1802, 8086, BAE RAD750, Microsemi SAMRH71 etc.);
- Delta-V engine for orbit changes (Apogee Kick Motor (AKM), Liquid Apogee Engine (LAE), etc.);
- Station-keeping thrusters (Rocket Engine Assembly (REA), Electrothermal Hydrazine Thruster (EHT)s, Arc jets, Ion, etc.);
- Interfaces (Analog-to-Digital (A/D) and Digital-to-Analog (D/A) converters, multiplexers, data buses, etc.).

You will note that many terms end with "assembly." This is because the device is a combination of electronics, optics, mechanisms, etc. that together form an assembly. This may just be RCA Astroelectronics nomenclature, but it is used throughout this book. Within each of these categories there are many choices. For example, star sensors can be trackers, cameras, or mappers and may have built-in star identification or may pass a pixel map to the spacecraft computer. Sun sensors can be digital or analog, have one or two axis of sensing and may be high or low accuracy. All of these items must be chosen in conjunction with the other subsystem engineers and the program-management office. Often the payload dictates the spacecraft type.

This section gives a qualitative discussion of attitude control system design. Its purpose is to outline an approach to designing a spacecraft control system and provide the reader with an introduction to some of the options.

4.2. Design flow

Fig. 4.1 shows a typical spacecraft-design flow with some options.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00015-2

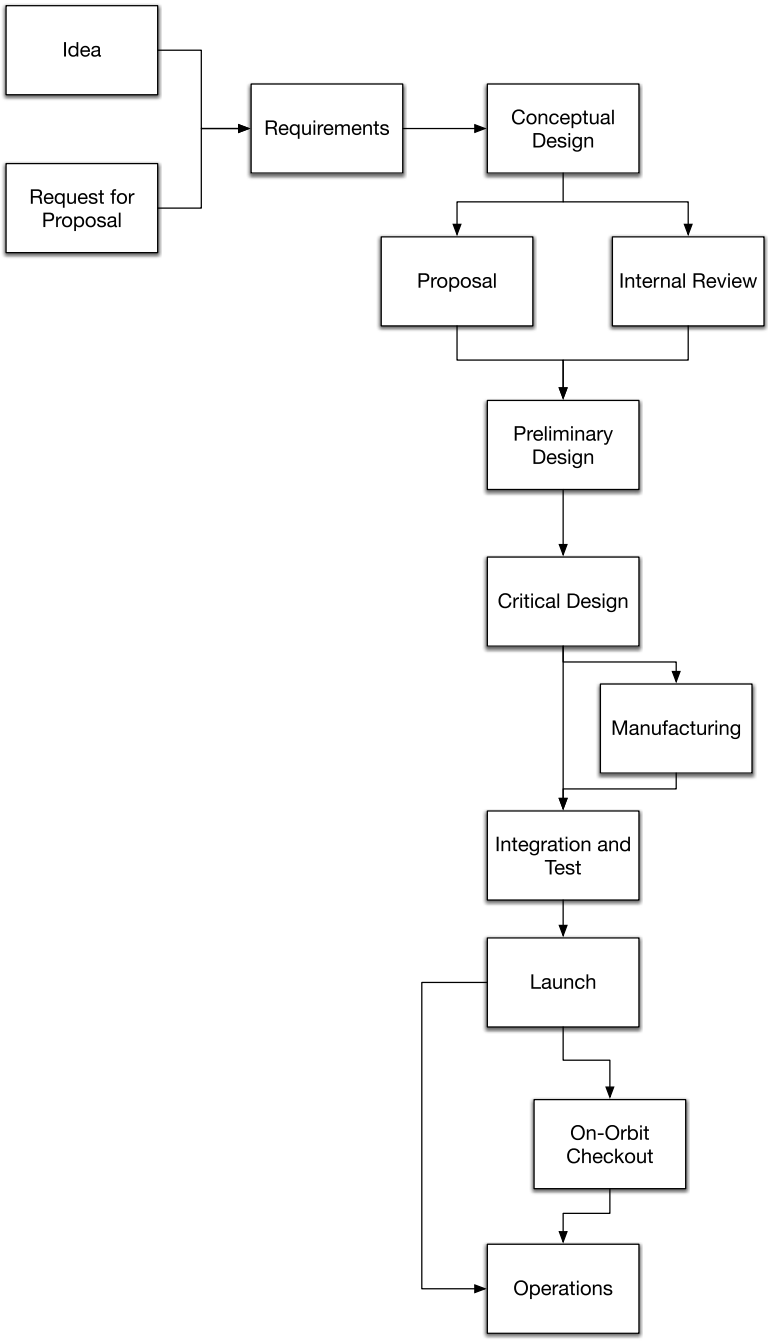


Figure 4.1 Spacecraft-design flow.

Table 4.1 gives system engineering and proposal terms. Not every spacecraft goes through these stages or has all of these reviews. Some may have more.

Table 4.1 System-engineering terms.

Term	Definition
CDR	Critical Design Review
LRR/FRR	Launch/Flight Readiness Review
MRR	Manufacturing Readiness Review
PDR	Preliminary Design Review
RFI	Request for Information that may lead to an RFP
RFP	Request For Proposal
PTR	Pink Team Review
RTR	Red Team Review

4.3. Organization of ACS design teams

An attitude control system consists of hardware, algorithms, and software. The algorithms are typically embedded in the software. They take data from sensors and process it to produce the commands to the actuators to accomplish the control objectives. Different engineers take on different roles in this process. Fig. 4.2 shows one decomposition of the roles for an ACS design. We distinguish between the ACS Controls Engineer and Systems Engineer. The former is responsible for doing all of the analysis and design of the ACS system. The latter does requirements analyses, interacts with the hardware vendors and program offices and also with the other subsystem engineers. Simulation may be a produce of the ACS analysts or be done by the software group. The software engineers implement the ACS algorithms in the flight software and write the interface software that connects the algorithms to the hardware. ACS engineers interact with propulsion if thrusters are used for attitude control. This diagram shows only the major roles and connections.

4.4. Requirements analysis

4.4.1 Direct requirements

Each mission will have a set of requirements. Requirements are either direct or derived. Typical direct mission requirements are for:

- Pointing accuracy;
- Pointing knowledge;
- Jitter;
- Lifetime;

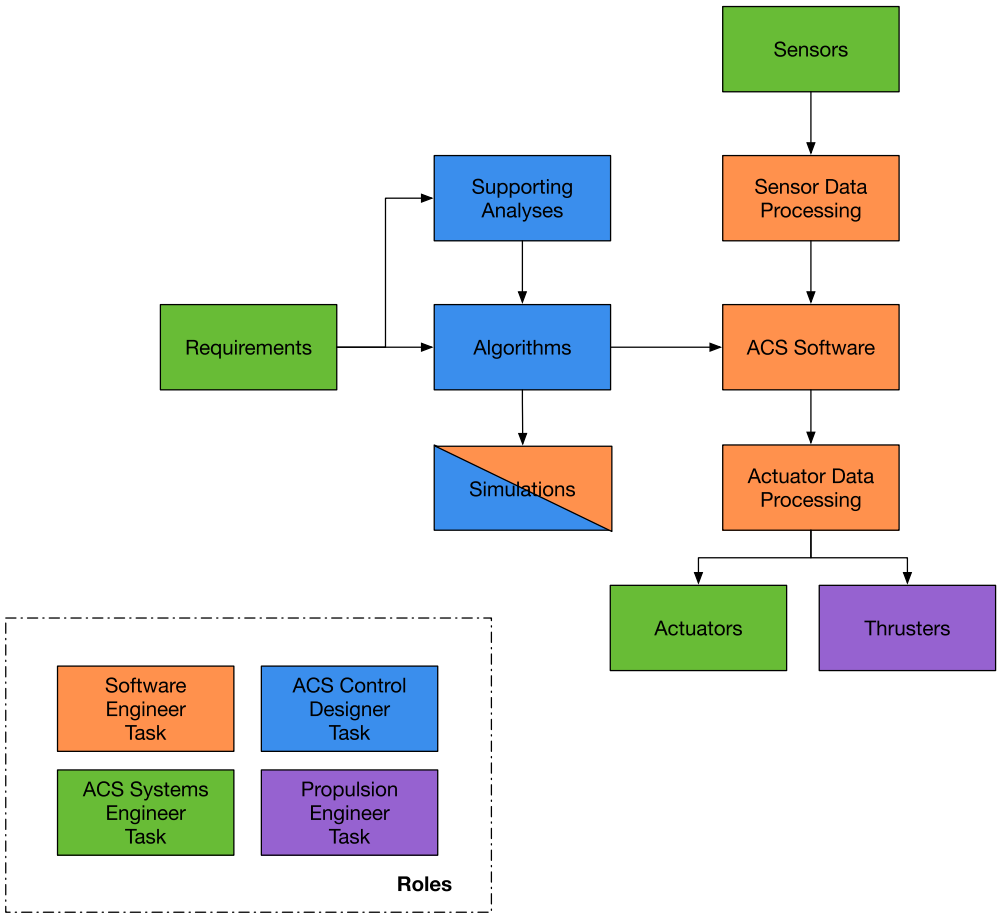


Figure 4.2 Spacecraft control-system design flow.

- Reliability;
- Redundancy.

Pointing accuracy is how tightly the spacecraft points at its target and knowledge is how well you know the attitude error. For example, a passively stabilized spacecraft with a star sensor might have very high attitude knowledge but relatively poor pointing accuracy. Jitter is measured in a variety of ways but ultimately it is a measure of how much the attitude changes in a given period of time that is usually not controllable by the pointing-control system. For example, if you use noisy sensors and low-resolution D/A (digital-to-analog) converters, you are likely to have a lot of jitter. Jitter is attitude motion that is not controllable by the control system. It may be beyond the bandwidth of the controller or hidden by a notch filter in the control loop.

The pointing numbers may be given for the spacecraft as a whole or for individual payloads. If the latter is the case, you have the additional option of putting some of the payloads on articulated platforms with their own sensors to improve the pointing of those payloads. For example, on a dual-spin spacecraft only the despun platform has a pointing requirement. The spinning part is only required to store momentum (and sometimes generate power). Pointing requirements only apply to the payload. It is not unusual to collocate the payload and attitude control sensors on the same composite structure (i.e., structurally stiff with low coefficients of thermal expansion) to minimize errors due to structural vibration and thermal distortion.

The lifetime requirement stipulates that the spacecraft must have a certain probability of lasting for the desired life-span. Two things can happen to terminate a mission. The spacecraft can run out of consumables (usually fuel but sometimes coolant for payloads), or the components can fail or wear out.

Redundancy requirements are often given as how long it takes the spacecraft to return to normal operation after a failure. The assumption is that no one failure will impact the mission, i.e., cause it not to meet its requirements, except for a permitted transient period after the failure. Different failures may have different requirements. Commercial customers usually require the quickest recovery from a failure since they lose money if the satellite is out of operation, not to mention irritating millions of customers whose sports broadcast is suddenly interrupted. Scientific customers are usually more lenient. The Hubble Space Telescope was operating out of specification for a long time before it was fixed.

The customer does not always specify the requirements carefully and may change them without notice as the design evolves. Thus, aside from meeting the baseline requirements, it is always necessary to make the design robust enough to tolerate requirements changes.

Another problem with requirements is that the customer may specify requirements in a way that needlessly constrains the design. For example, a geosynchronous-satellite customer may specify the Euler angles for roll, pitch, and yaw (used to measure spacecraft orientation) when they really mean to specify beam pointing (circular error) or even better, power distributed over the coverage area. If you are given the third type of requirement you could meet it by increasing the beam power. If you are given the second you could trade roll accuracy against pitch accuracy and ignore yaw altogether (with the caveat mentioned above).

4.4.2 Indirect (or derived) requirements

Derived requirements are levied on the ACS subsystem by other subsystems. If the power over the coverage area is the direct requirement then attitude accuracy becomes an indirect requirement. Power consumption of your components is always an indirect requirement. Some indirect requirements are:

- Power;
- Thermal;
- Radiation hardness;
- Radio-frequency emission sensitivity;
- Induced noise (such as from an unbalanced momentum wheel);
- Computational loading (i.e., how many floating-point operations you can use per control cycle).

4.4.3 Control-system requirements

Traditional control-system requirements, such as gain and phase margins, are rarely specified. They must be derived from the pointing and jitter requirements. For example, most satellites have a special thruster control mode. When you turn on the delta-V thrusters, there will be an attitude transient. This transient must not exceed the pointing requirements for the spacecraft; hence this puts a limit on the peak overshoot that is permissible. It is usually desirable not to have to change control-system gains over the mission life; hence this places a requirement on the tolerance of the control system to properties changes. Sensor-scale factors may drift due to accumulated radiation damage, leading to a requirement on the tolerance to scale-factor variations.

While some, such as sensor- and actuator-scale factors and inertia variations, can be lumped into an aggregate gain variation, it is a good idea to perform a sensitivity analysis on the closed-loop control system for all identifiable parameters that are known to vary.

4.5. Satellite design

4.5.1 Selecting a satellite configuration

Satellites generally have two major phases in their life. The first phase is transfer orbit during which the satellite is placed in its operational orbit. The second phase is the mission orbit where the satellite performs its nominal mission. Deep-space spacecraft may have long transfers. They often have separate scientific requirements during transfer to their final target. For example, New Horizons had requirements for its Pluto flyby, but also has requirements for its flight into interstellar space. Low-Earth orbiting satellites are often placed directly into their mission orbits and do not have a transfer-orbit phase.

The first step is choosing the mission-orbit configuration, followed by choosing the transfer-orbit configuration. A few of the more popular satellite configurations are:

- Gravity-gradient stabilized;
- Spinner;
- Dual-spin;
- Bias momentum;
- Zero momentum;
- Sun-nadir.

Gravity-gradient stabilization is passive and at most requires a libration damper. Libration dampers are dampers designed to damp spacecraft rigid-body oscillations due to gravity-gradient disturbances. If there is sufficient internal damping (due to heat pipes, fuel slosh, or structural damping), even the damper may not be required. Gravity-gradient stabilization is suitable only for spacecraft with loose pointing requirements that do not have to change their orientation.

A spinner can be used if one axis of the spacecraft is required to remain inertially fixed and the payload is not affected by spinning, for example a satellite with a long wire antenna along the spin axis. If the satellite spins about its major axis a passive damping system will suffice. If it has to spin about its minor axis an active nutation control system (to damp the natural oscillation of a spinning spacecraft at the nutation frequency) is needed to keep it out of a flat spin (i.e., a steady spin about any other axis). The satellite will also need a control system to keep the spin axis in the same inertial direction if there are significant inertially fixed torques, for example due to a center-of-solar-pressure offset from the center-of-mass along the spin axis.

Dual-spin spacecraft have two rotating parts. Usually one is a platform that rotates at orbit rate (and consequently is nominally pointing at the Earth). The other rotates at high speed and gives the spacecraft gyroscopic stiffness. A dual-spin spacecraft has a large structure on the spacecraft rotating. Often the solar cells are attached to the rotating part.

A bias-momentum spacecraft is conceptually the same as a dual-spin spacecraft, except that the part that is rotating at high speed is a momentum wheel and it rotates at very high rates, on the order of 6000 rpm. Its momentum may be large but the wheel's spin-axis moment of inertia is small. Both dual-spin and momentum-bias spacecraft have been used historically to keep costs down for commercial communication satellites since the gyroscopic stiffness helps to keep the spacecraft oriented. The coupling between the roll and yaw axes allows the elimination of the yaw sensor, which was the most expensive on the spacecraft since it requires sensing of the Sun.

A zero-momentum design has, ideally, zero net inertial momentum. The spacecraft may be controlled by reaction wheels, control-moment gyros, magnetic torquers, or by thrusters. With a spacecraft with momentum-exchange devices, they may each have significant momentum but the vector sum of their momentum is zero. It is relatively easy to rotate a spacecraft with zero momentum because there is no gyroscopic stiffness to resist rotation. However, if the control system fails, there is also no gyroscopic stiffness to keep the spacecraft from rotating and it may tumble. Magnetic torquers can only control two axes at a time but over an orbit can provide "average" three-axis control.

A Sun-nadir pointing spacecraft is a spacecraft that points the bus at nadir and the solar-array normal at the Sun. This type of spacecraft has only one degree of rotational freedom for the solar array. Thus to keep the arrays on the Sun the spacecraft must rotate about the axis pointing towards the Earth (yaw). This allows the spacecraft to produce more power than would be otherwise possible.

4.5.2 Selecting a control strategy

Passive control can be used on simple satellites that do not have tight pointing requirements. One strategy would be a gravity-gradient-stabilized satellite with a ball-in-tube libration damper. The damper damps out libration frequencies (the frequency of the gravity-gradient modes of oscillation). Active control requires either analog or digital electronics, sensors, and actuators and can add greatly to the cost of a satellite.

4.5.3 Selecting actuators

Typical satellite actuators are:

- Thrusters;
- Gimbaled thrusters;
- Wheels or spheres (momentum or reaction);
- Pivoted wheels (control-moment gyros, for example);
- Magnetic torquers;
- Reflective surfaces (either to solar or aerodynamic drag).

We will discuss thrusters, wheels, and pivoted wheels further in this section. Additional details may be found in Chapter 10.

4.5.4 Thrusters

If the satellite has a requirement to maintain its orbit it will require thrusters and a complete propulsion system. It is almost always necessary to use some thrusters for attitude control during orbit-adjustment maneuvers. This may be done by firing other thrusters to provide control torques to compensate for the disturbance torques generated by the delta-V thrusters, or by gimbaling the delta-V thrusters. For example, the Space Shuttle Orbiter does both. It gimbals the Orbital Maneuvering System (OMS) engines and also fires bipropellant thrusters. Gimbaling is generally not very popular because the gimbal mechanisms tend to be heavy, complex, and expensive. A gimbal was used on NASA's Deep Space 1 to point the ion engine through the center-of-mass, effectively providing 2-axis control.

Thrusters can be throttled, fired with a fixed-length pulse, or pulsewidth modulated. Pulsewidth modulation means that the thruster is on at a fixed thrust level for only part of the control period so that the average torque equals the commanded torque. This technique can be extended to have a variable firing period (known as pulsewidth–pulse-frequency modulation). Throttled thrusters, which can produce a variable magnitude of thrust, tend to be expensive and complex and are rarely used. Pulsewidth modulation is frequently used because it lends itself to linear control systems. Fixed pulsewidth-control systems are derived from optimal control theory and minimize fuel consumption. They are used on the Shuttle and were used on the Apollo spacecraft. However, they do not lend themselves to spacecraft that cannot be modeled as rigid bodies.

4.5.5 Wheels

A momentum wheel nominally has a fixed (large) momentum. A reaction wheel nominally has zero momentum. A reaction wheel (or reaction sphere) works by providing a reaction to the torque that the motor produces. The torque is applied equally to the spacecraft and to the wheel. Wheels present an interesting control problem. As they spin up they create a momentum bias on the spacecraft, thus changing the system dynamics. This perturbation is nonlinear because the torque on the spacecraft is the product of the spacecraft angular velocity and the wheel momentum. If the disturbances are small, and mostly cyclic, and the spacecraft has a relatively large inertia, the plant perturbation is second order. Many algorithms have been written to take advantage of this during maneuvers. The algorithm will try to pick a path in velocity space that minimizes time while staying within the momentum limits of the wheels.

4.5.6 Pivoted wheels and control-moment gyros

Pivoted wheels use pivots (either one or two) to generate transverse axis torques. They also use the wheel like a reaction wheel. Control-moment gyros (CMGs) use only the gimbal motors to provide torque and the wheel is maintained at a constant speed. CMGs come in double- and single-gimbaled types. The former require complex slip rings to get power to the inner gimbal. Since one CMG only provides at most two axes of torque, at least two are required. Generally, more are needed and it is necessary to distribute the control activity among the gimbals in an optimal way (usually to keep the CMG platforms at small angles relative to the spacecraft). CMGs are popular for maneuvering spacecraft of intermediate size or for really large spacecraft (like Skylab, the first American space station, the International Space Station, or certain Earth-observing systems).

A single pivoted wheel is shown in Fig. 4.3. Pivoted wheels are used on Lockheed Martin momentum-bias communications satellites and on the Hughes HS601 spacecraft. The former uses a single pivot about roll while the latter employs a double pivot. Pivoted wheels and CMGs require momentum unloading unless the complementary filter approach is used. In either case, an actuator that can apply an external torque to the spacecraft, such as a thruster or magnetic torquer, is required. Reaction wheels, pivoted wheels, and CMGs can redistribute momentum within a spacecraft; they cannot change the total inertial momentum of the spacecraft or change its vector in the inertial frame. Many spacecraft with station-keeping requirements only unload momentum during stationkeeping maneuvers when it can be done "for free" using the station-keeping thrusters.

4.5.7 Selecting sensors

Typical satellite sensors are:

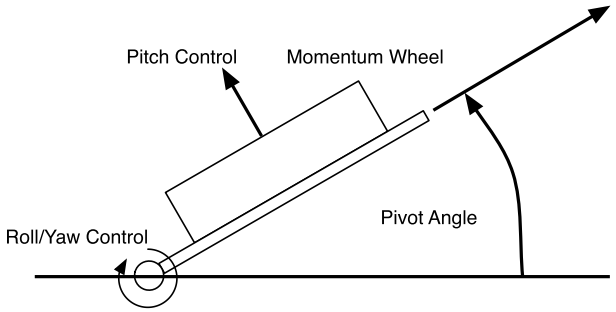


Figure 4.3 Pivoted wheel.

- Earth sensors (scanning or static);
- Star sensors (mappers or trackers);
- Sun sensors (one or two axis);
- Horizon sensors (need a spinning spacecraft);
- Gyros (many types);
- Magnetometers;
- Potentiometers;
- Angle encoders.

Many satellites are Earth or planet pointing and need to point a payload at a target on the Earth or a planet. Some are inertially pointing (such as the space telescope) and point to targets away from the Earth. Star sensors and Sun sensors give an inertial reference directly and can, through means of transformation matrix from the inertial to planet fixed frame, give an Earth reference. Earth sensors, magnetometers, and horizon sensors give an Earth-fixed reference, but are much less accurate than star and Sun sensors.

Gyros measure inertial rotation rates. Rate-integrating gyros give the integral of the body rates. Since each gyro integrates about a body axis, gyros cannot give an inertial reference directly and are always used in conjunction with magnetometers, star, Sun, or Earth sensors.

Angle encoders and potentiometers are used to measure relative orientations of components on the spacecraft. Encoders have potentially very high resolution but are expensive. Potentiometers, resistors whose output varies with angle, are generally cheaper than encoders and are often used for "sanity checks" on components driven by stepping motors. The stepping-motor position is usually known from counting steps but it is possible to lose count, for example at spacecraft separation from the launch vehicle.

Selection of sensors is a process of trading cost against sensor life, accuracy, mass, power consumption, and processing-software complexity. An important factor in selecting sensors is their radiation hardness. Radiation will degrade many analog circuits

and most CCD, CMOS, or CID imaging sensors, thus limiting the life of optical sensors. Other considerations include sensor placement and interference with the payload.

4.5.8 Selecting processors

The four major requirements for processors are

1. Processing throughput;
2. Memory;
3. Radiation hardness;
4. Heritage.

The processor needs to have sufficient throughput to perform all the computations required by the spacecraft. It needs to have sufficient memory for all required software. For example, the original processors on the Space Shuttle did not have enough memory for all of a missions' software, so it was necessary to load in new software during the mission. Radiation hardness is a requirement for all but very low-Earth orbiting satellites. Criteria are both the total-dose rating and immunity to single-event upsets (SEU) and latch-up immunity. The latter is when the processor freezes and has to be rebooted.

The last criteria, heritage, is important since it is much less expensive to use a processor that you have already used before. Changing processors often means changing software-development tools, operating systems, and even languages. These kind of changes have a major impact on the cost and schedule of a satellite program.

4.5.9 Selecting delta-V engines

The selection of the delta-V engine has an impact on the mission time line, the spacecraft cost, and all of the subsystems on the spacecraft. By delta-V engine we usually mean engines that are specifically designed for orbit-change operations and not for attitude control. The three major types are

1. Solid motor;
2. Liquid motor;
3. Electric motor.

The first gets the spacecraft into orbit quickest but usually results in the largest orbital errors that must be corrected with other thrusters. A solid motor is usually large and requires substantial structure. The liquid motor usually requires several burns for orbit raising. This means that transfer-orbit operations take longer but the final orbit is more accurate. Also, liquid motors are more efficient than solid motors thus lowering the overall mass of the vehicle and reducing launch costs. Electric motors are the most efficient but require high power for high thrust and still may take months to put the spacecraft in its final orbit.

4.5.10 Selecting the station-keeping engines

If station keeping is required station-keeping rocket motors are needed. If large amounts must be done (e.g., for inclination control of a geosynchronous satellite requiring about 500 m/s for 10 years) then an electric motor may make sense. For small amounts, conventional bipropellant or monopropellant thrusters would be sufficient. For very precise station keeping small electric thrusters can be used.

4.5.11 Selecting the interfaces

If you have a flight computer you need to hook it into your sensors and actuators. A spacecraft might have hundreds of sensors and actuators when you include the thermal-control subsystem. The data rates for each device vary. A heater requires a single bit to turn it on or off but a camera might be sending a megabit each second for processing. Typical serial interfaces are

- Analog lines multiplexed to a common A/D converter;
- I2C a popular serial bus;
- Databus (e.g., MIL-STD-1553, CAN, SpaceWire, Ethernet);
- RS-232 - a 1960s standard for the serial communication of data;
- RS-422 - a replacement for RS-232 permitting higher speeds and with better noise immunity.

Most spacecraft have a combination of interfaces to accommodate specific hardware. For example, many sensors and actuators only provide RS-422 interfaces, so an all-SpaceWire architecture may not be feasible. It is common to have an entire circuit board dedicated to analog input and output lines. Many popular processors, such as Arduino, have numerous analog pins.

Parallel interfaces are also possible, such as direct memory access (DMA) or IEEE 1284. For example a camera may benefit from a parallel interface to the flight processor. DMA allows a device to access the computer memory directly, thus saving on processor overhead.

4.5.12 Cost

Most manufacturers wish to design a satellite that meets all requirements and costs as little as possible (unless they have a cost-plus, fixed-fee contract). The total cost of a spacecraft is composed of recurring and nonrecurring costs. Recurring costs are the costs associated with building each spacecraft. Nonrecurring costs are costs associated with designing a spacecraft or customizing an existing design. Each type of cost is composed of the cost of personnel and the cost of purchased products, facilities, and other nonpersonnel costs. Recurring costs are discussed in Table 4.2. Nonrecurring costs are described in Table 4.3.

Table 4.2 Recurring costs.

Cost	Description
Direct Cost of Components	Costs of components generally rise with the required lifetime of the component and the device's performance. The cost of integrating a component into a design must be added to this cost. Sometimes, a more expensive component is much less expensive to integrate.
Direct Cost of the Consumables	The cost of consumables (such as fuel and cryogenics) must include the cost of loading the consumable and the cost of storing the consumable onboard the spacecraft while it is on the ground.
Cost to Lift the System into Orbit	Mass is almost always at a premium on any spacecraft. The transportation cost must always be considered. The greater the required delta-V for the mission the higher the cost. This is more complicated than it seems at first since there are thresholds of mass where you must go to the next size up launch vehicle (like from a Delta to an Ariane) that then has a different price schedule. This cost can be lowered by sharing a launch vehicle or going up on a test launch. The latter was used by PanAmSat to get a free launch and get started in the commercial communications-satellite business.
Cost of Insurance	New designs cost more to insure than proven designs. The cost of insurance also varies with the launch vehicle. Insurance is quite expensive and is provided by specialized insurance companies.
Cost of Maintenance	Technical support for a customer can be very expensive, especially if it requires onsite support. In addition to the direct cost of the engineering support, the cost of losing the services of the engineers as they move to other assignments, or leave the company, must be considered. Training of new engineers to support spacecraft is rarely done in a methodical way and can lead to support problems.

Table 4.3 Nonrecurring costs.

Cost	Description
Cost to Design the Algorithms	This includes the design costs and the cost of all tools needed to generate the design.
Cost to Implement in Software	Includes software requirements generation, software design, coding, and test.
Cost to Document the System	Includes memos, reports, and user guides.

CHAPTER 5

Kinematics

5.1. Space story

One of the challenges for a momentum-bias communication is to reorient it from its spinning transfer-orbit orientation to its mission orientation. In transfer orbit, it is a spinner and the momentum wheels are fixed. In mission orbit, the momentum-wheel axis is normal to orbit normal and the spacecraft is rotating at orbit rate, that is one revolution per day in geosynchronous orbit. The obvious approach is to despin the spacecraft with thrusters, reorient it with thrusters and then acquire the Earth. This would require three-axis sensing. Dr. Carl Hubert of RCA Astro-Electronics came up with the idea of the dual-spin turn. After the burn that puts the spacecraft in orbit, you align the spin axis with orbit normal. The same sensing can be used that is used for all of the transfer orbit. You spin down to the right angular rate so that the total angular momentum is what you want in the momentum wheel. You then turn on the momentum wheel. Angular-momentum conservation ensures that the momentum-wheel vector will align itself with orbit normal. Some damping is required. A patent was filed for major-axis spinners that included damping. However, it turned out that if the spacecraft had fuel, and any kind of mechanism to damp slosh in the tanks, then fuel slosh provided sufficient damping so that no additional damping was needed. This provided a mechanism for other satellite manufacturers to use the dual-spin-turn without a patent license. Dr. Hubert later patented a version that allowed you to do the dual-spin-turn with the starting spin about the minor inertia axis.

5.2. Introduction

Given two frames, the orientation of one frame with respect to the other can be defined in several ways. Three that are widely used are quaternions, Euler angles, and transformation matrices. A quaternion is a four-element vector incorporating an axis and angle of rotation. Euler angles are a series of three rotations. A transformation matrix is an orthonormal matrix that transforms a vector measured in one frame to another frame. An orthonormal matrix is a matrix whose inverse equals its transpose and for which each column or row is a unit vector. Fig. 5.1 shows a vector that could be measured in either of two frames.

The vector orientation and length do not change, but its x, y, and z components change depending on which frame is used to measure it. Kinematics in spacecraft refers to the expression and propagation of the satellite attitude, using one of these represen-

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00016-4

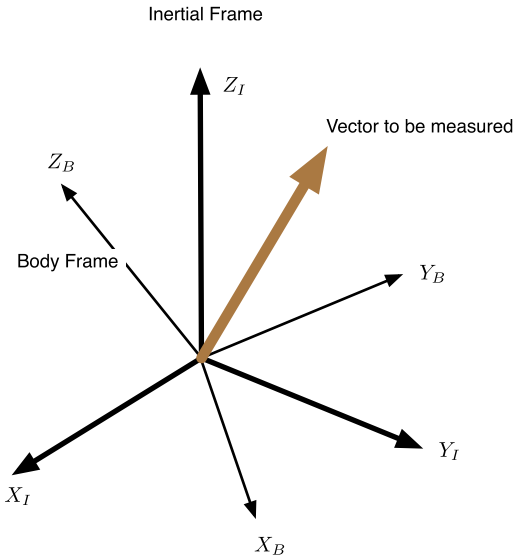


Figure 5.1 Two-frame vector diagram.

tations, without considering any accelerations. The effect of accelerations is considered in the next chapter, Dynamics.

5.3. Euler angles

Euler angles are commonly used to represent attitude or orientation. Fig. 5.2 shows a 3-2-1 Euler angle set. The first rotation is about the 3-axis of both the inertial and body frames. The second is about the 2-axis of the body frame and the third is about the 1-axis of the body frame.

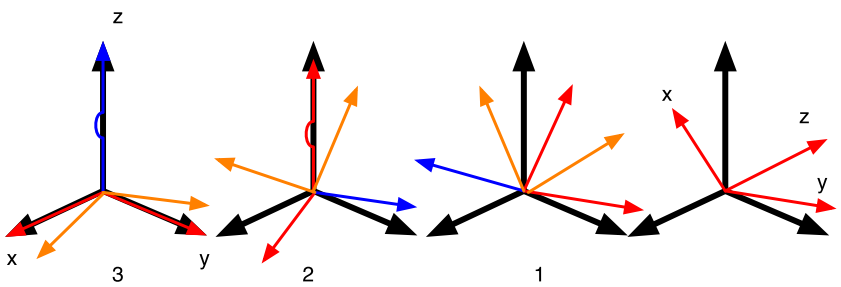


Figure 5.2 Euler-angle diagram.

Each rotation can be used to define a transformation matrix. The product of three transformation matrices gives the overall transformation matrix.

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ 0 & -\sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 \\ 0 & 1 & 0 \\ \sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \begin{bmatrix} \cos\theta_3 & \sin\theta_3 & 0 \\ -\sin\theta_3 & \cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

There are twelve sets of Euler angles. Each set of three rotations is capable of pointing a unit vector in any orientation. Alternately, M could be obtained by a single rotation about a unit vector. The combination of rotation angle and unit vector gives the quaternion.

It is generally not recommended to propagate Euler angles. If you propagate Euler angles the propagator must compute numerous trigonometric functions, which slows down the propagation. In addition, there are singularities where only two axes of rotation are defined. It is much simpler (and computationally more efficient) to propagate quaternions (Section 5.5).

5.4. Transformation matrices

If u_b is the vector in the body frame, and u_I is the vector as measured in the inertial frame, the two are related by

$$u_I = M_{BToI} u_b \tag{5.2}$$

Note that

$$M_{IToB} = M_{BToI}^T \tag{5.3}$$

The transformation matrix is defined as

$$M_{BToI} = \begin{bmatrix} x_{BInI}^T \\ y_{BInI}^T \\ z_{BInI}^T \end{bmatrix} \tag{5.4}$$

where each row is the unit vector of the axis of the body (B) frame as measured in the inertial (I) frame.

Fig. 5.3 shows two coordinate frames. Each of the axes is represented by a unit vector such that

$$Z \times X = Y \tag{5.5}$$

$$X \times Y = Z \tag{5.6}$$

$$z \times x = y \tag{5.7}$$

$$x \times y = z \tag{5.8}$$

This implies that the coordinate frames are orthogonal, i.e., every axis makes a right angle with every other axis.

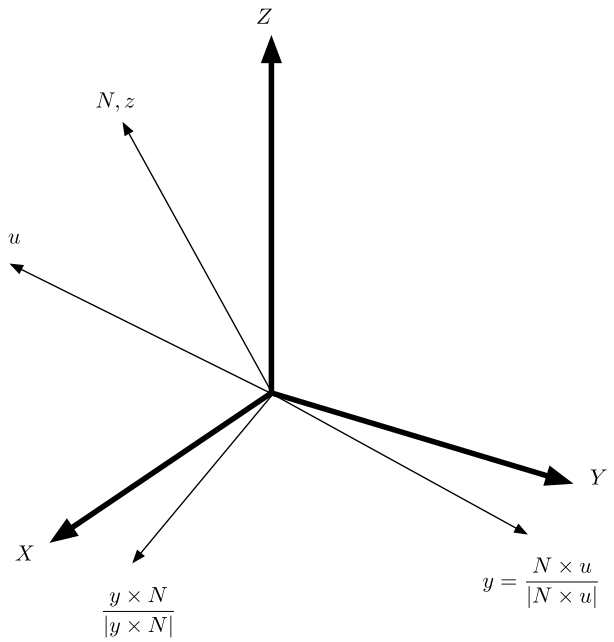


Figure 5.3 Transformation matrix from two vectors.

The vector a can be parameterized using coordinates from either set of axes. If x, y, and z are parameterized as

$$x(X, Y, Z) \tag{5.9}$$

$$y(X, Y, Z) \tag{5.10}$$

$$z(X, Y, Z) \tag{5.11}$$

then

$$a(x, y, z) = \begin{bmatrix} x(X, Y, Z) \\ y(X, Y, Z) \\ z(X, Y, Z) \end{bmatrix} a(X, Y, Z) \tag{5.12}$$

The x component of a in the (x, y, z) system is just the dot product of the x-axis, $x(X, Y, Z)$ with $a(X, Y, Z)$.

As the axes are orthogonal unit vectors, the following identity holds

$$E = CC^T \tag{5.13}$$

where C is the transformation matrix, and E is the identity matrix. The transformation matrix from (x, y, z) to (X, Y, Z) is just C^T. That is $C^T = C^{-1}$. Often it is necessary to create a transformation matrix from two vectors. The following gives an example. Two vectors are parameterized in (X, Y, Z), N and u. Define N to be the z-axis. Then, the axes are

$$z = \frac{N}{|N|} \tag{5.14}$$

$$y = \frac{N \times u}{|N \times u|} \tag{5.15}$$

$$x = \frac{y \times z}{|y \times z|} \tag{5.16}$$

where x, y, and z are orthonormal and unit vectors.

Multiple transformations are performed by multiplying a vector by each matrix in succession.

$$u_c = C_{cb} C_{ba} u_a$$

where the notation C_{ba} is equivalent to writing $C_{\text{a to b}}$ and the inner b frames visibly cancel out, $C_{ca} = C_{cb} C_{ba}$. An example of a sequence of rotations requiring this notation is a set of gimbals.

5.5. Quaternions

5.5.1 Introduction

A quaternion is a convenient way of representing the orientation of one frame with respect to another. Propagation of the orientation of an object with quaternions is efficient since only four elements are propagated and no trigonometric functions are needed. In addition, it does not have any singularities.

Euler angles are useful for visualizing the attitude. Quaternions should be used for numerical integration. If more than three vectors are to be transformed from one frame to another using the same quaternion, it is faster to first convert the quaternion to a rotation matrix and then do the matrix–vector multipliers to get the transformations. Transforming a vector directly with a quaternion takes 30 floating-point operations; it takes 45 operations to convert from a quaternion to a matrix. A matrix transformation takes 15 operations.

An example of a quaternion is shown in Fig. 5.4. Assume that we have two reference frames that have the same z-axis and a vector u that can be represented in either frame. The length of u is the same in each frame. In this case, the z component of u will be the same in both frames A and B but the x and y components will have different values. It is important to be precise about the definition of a quaternion. Specifically, q_{ATOB}

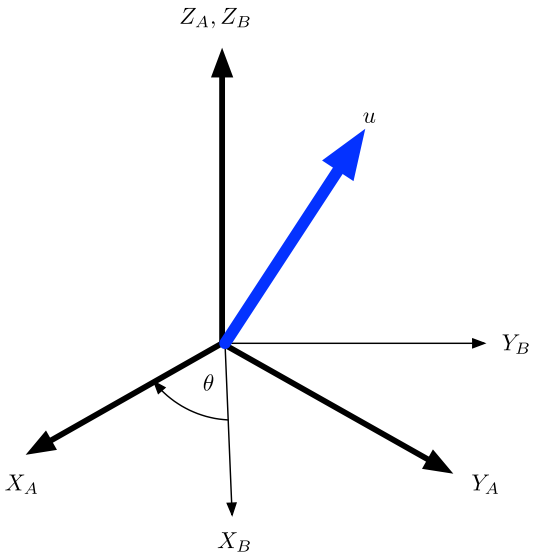


Figure 5.4 Quaternion diagram.

transforms a vector represented in the A-frame coordinates into a vector represented by the B-frame coordinates. This same quaternion might be notated instead as q_{BA}, where the order of the subscripts now indicates that the quaternion transforms a vector in frame A to a vector in frame B. This form can be helpful when combining quaternions in order as discussed below. Both are used in practice and it is important to always verify the rotation represented.

5.5.2 Fundamental properties of the quaternion

A quaternion is a four-parameter set used to describe the orientation of one reference frame with respect to a second reference frame. Although only three parameters are needed to uniquely specify the relative orientation, all three-parameter sets have singularities that make them unsuitable for numerical simulations, or integration inflight software. Four parameter sets, such as quaternions, do not have problems with singularities. The quaternion is represented by a four row-vector

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} s \\ v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\phi}{2} \\ a_1 \sin\frac{\phi}{2} \\ a_2 \sin\frac{\phi}{2} \\ a_3 \sin\frac{\phi}{2} \end{bmatrix} = q \tag{5.17}$$

The first notation is the standard used in the software. No distinction is made between the four elements of the quaternion since numerically there is none.

Occasionally, authors will make the first element q_0 and the remainder $[q_1\ q_2\ q_3]^T$ to correspond to the scalar and vector notation. The second notation in (5.17) breaks the quaternion into a scalar and 3-vector part. Letting the first term be the scalar is arbitrary.

The third form relates to the theorem of Euler that any arbitrary rotation about any number of axes can be reduced to a single rotation about a fixed axis. If the rotation angle is defined as ϕ and the axis of rotation is a, then the sign of the first value is arbitrary since any rotation can be represented by two quaternions of opposite sign. In this text, the first element will always be positive. This would imply that ϕ will always be between $\pm 180°$. The transpose of a quaternion is defined as

$$q^T = \begin{bmatrix} q_0 \\ -q_1 \\ -q_2 \\ -q_3 \end{bmatrix} = \begin{bmatrix} s \\ -v_1 \\ -v_2 \\ -v_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\phi}{2} \\ -a_1 \sin\frac{\phi}{2} \\ -a_2 \sin\frac{\phi}{2} \\ -a_3 \sin\frac{\phi}{2} \end{bmatrix} \tag{5.18}$$

with the property that

$$qq^T = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.19}$$

This notation comes from the NASA Space Shuttle world. That is, the unit quaternion has a 1 as the first element. In many organizations the unit quaternion is

$$qq^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.20}$$

The NASA Jet Propulsion Laboratory (JPL) uses this convention. Different companies may use different conventions.

5.5.3 Quaternion nomenclature

Quaternions can be written as

$$q_{ab} \tag{5.21}$$

which is defined as the quaternion that rotates a vector from frame b to frame a (right to left in the order of the coefficients). Thus

$$u_a = q_{ab} \otimes u_b \tag{5.22}$$

and quaternion multiplication is defined as

$$q_{ac} = q_{ab} \otimes q_{bc} \tag{5.23}$$

Quaternion multiplication is defined as $\otimes$, which turns the first 4-by-1 quaternion vector into a 4-by-4 matrix that then is multiplied like any matrix.

The inner frames cancel in the multiplication. This is consistent with the notation often used for transformation matrices. Quaternion multiplication will be discussed in the next section. In the following sections, quaternions will not use the full subscript notation for compactness.

5.5.4 Quaternion operations

Expanding quaternion multiplication

$$q_1 \otimes q_2 = \begin{bmatrix} s_1 s_2 - \vec{v}_1 \vec{v}_2 \\ s_1 \vec{v}_1 + s_2 \vec{v}_2 + \vec{v}_1 \times \vec{v}_2 \end{bmatrix} \tag{5.24}$$

where s_1 is the scalar part of the first quaternion and s_2 is the scalar part of the second quaternion, $\vec{v}_1$ and $\vec{v}_2$ are the corresponding vector parts. The arrows are used to clarify that they are vectors. Quaternion multiplication can be expressed in matrix form in two ways

$$q_1 \otimes q_2 = \begin{bmatrix} q_1(1) & -q1(2) & -q_1(3) & -q_1(4) \\ q_1(2) & q1(1) & -q_1(4) & q_1(3) \\ q_1(3) & q1(4) & q_1(1) & -q_1(2) \\ q_1(4) & -q1(3) & q_1(2) & q_1(1) \end{bmatrix} q_2 \tag{5.25}$$

or

$$q_1 \otimes q_2 = \begin{bmatrix} q_2(1) & -q_2(2) & -q_2(3) & -q_2(4) \\ q_2(2) & q_2(1) & q_2(4) & -q_2(3) \\ q_2(3) & -q_2(4) & q_2(1) & q_2(2) \\ q_2(4) & q_2(3) & -q_2(2) & q_2(1) \end{bmatrix} q_1 \tag{5.26}$$

The numbers in the parentheses are the quaternion element number. The only difference between the equations is that the sign is reversed on the skew-symmetric 3×3 matrix in the lower right-hand corner. Once one quaternion is expanded into a four-by-four matrix, regular matrix operations apply.

5.5.5 Quaternion transformations

Quaternions transform vectors using the following operation

$$x_b = q_{ba} \otimes x_a \otimes q_{ba}^T \tag{5.27}$$

using quaternion multiplication with the vectors defined as quaternions with a scalar part equal to zero or

$$x_a = \begin{bmatrix} 0 \\ x_a(1) \\ x_a(2) \\ x_a(3) \end{bmatrix} \tag{5.28}$$

Writing this out in full

$$\begin{aligned} x_b = & \begin{bmatrix} q_{ba}(1) & -q_{ba}(2) & -q_{ba}(3) & -q_{ba}(4) \\ q_{ba}(2) & q_{ba}(1) & q_{ba}(4) & -q_{ba}(3) \\ q_{ba}(3) & -q_{ba}(4) & q_{ba}(1) & q_{ba}(2) \\ q_{ba}(4) & q_{ba}(3) & -q_{ba}(2) & q_{ba}(1) \end{bmatrix} \\ & \begin{bmatrix} 0 & -x_a(1) & -x_a(2) & -x_a(3) \\ x_a(1) & 0 & x_a(3) & -x_a(2) \\ x_a(2) & -x_a(3) & 0 & x_a(1) \\ x_a(3) & x_a(2) & -x_a(1) & 0 \end{bmatrix} \\ & \begin{bmatrix} q_{ba}(1) \\ -q_{ba}(2) \\ -q_{ba}(3) \\ -q_{ba}(4) \end{bmatrix} \end{aligned} \tag{5.29}$$

The middle matrix is the skew-symmetric matrix for Eq. (5.28).

5.5.6 Quaternion derivative

The derivative of a quaternion is defined as

$$\dot{q} = \lim_{\Delta t \to 0} \frac{\Delta q}{\Delta t} = \lim_{\Delta t \to 0} \frac{q(t + \Delta t) - q(t)}{\Delta t} \tag{5.30}$$

The derivative of the quaternion will be related to the angular velocity of one frame with respect to the other. There are four possible combinations for the derivative of a quaternion. They are

$$\begin{matrix} q_{ab}, \omega_a & q_{ba}, \omega_b \\ q_{ba}, \omega_a & q_{ab}, \omega_b \end{matrix} \tag{5.31}$$

The combinations are the direction of the quaternion rotation and the reference frame for the angular-rate vector. The quaternion can transform in either direction between a and b and the angular rate can be expressed either in frame a or b. Define the delta quaternion as

$$q(t + \Delta t) = \Delta q q \tag{5.32}$$

The new quaternion is just the old multiplied by a quaternion that transforms the first into the second. Write the angular rate as the product of a rotation axis, a, and a scalar angular rate

$$\vec{\omega} = a\omega \tag{5.33}$$

where a is the unit vector for $\vec{\omega}$. The derivations for each of the four sets are straightforward. Start with set q_{ba}, ω_b. Δq arises when expressed in the scalar-vector form of the quaternion,

$$\Delta q = \begin{bmatrix} \cos\frac{\omega\Delta t}{2} \\ a\sin\frac{\omega\Delta t}{2} \end{bmatrix} \tag{5.34}$$

and (5.30) becomes

$$\dot{q} = \lim_{\Delta t\to 0}\frac{\Delta q}{\Delta t} = \lim_{\Delta t\to 0}\frac{(\Delta q - e)q}{\Delta t} \tag{5.35}$$

where e is the identity quaternion

$$e = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.36}$$

Expressing Δq as the state-transition matrix for a quaternion it is

$$\Delta q = \begin{bmatrix} \cos(\frac{\omega\Delta t}{2}) & -a^T\sin(\frac{\omega\Delta t}{2}) \\ a\sin(\frac{\omega\Delta t}{2}) & E\cos(\frac{\omega\Delta t}{2}) - a^\times\sin(\frac{\omega\Delta t}{2}) \end{bmatrix} \tag{5.37}$$

where E is a 3-by-3 identity matrix.

$$E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.38}$$

and $a^\times$ is

$$a^\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{5.39}$$

then (5.35) becomes

$$q = \lim_{\Delta t\to 0}\frac{1}{\Delta t}\begin{bmatrix} \cos(\frac{\omega\Delta t}{2}) - 1 & -a^T\sin(\frac{\omega\Delta t}{2}) \\ a\sin(\frac{\omega\Delta t}{2}) & E\left(\cos(\frac{\omega\Delta t}{2}) - 1\right) - a^\times\sin(\frac{\omega\Delta t}{2}) \end{bmatrix} q \tag{5.40}$$

In the limit, the cosine terms become one, and the sine terms are their argument, $\sin(x) \approx x$

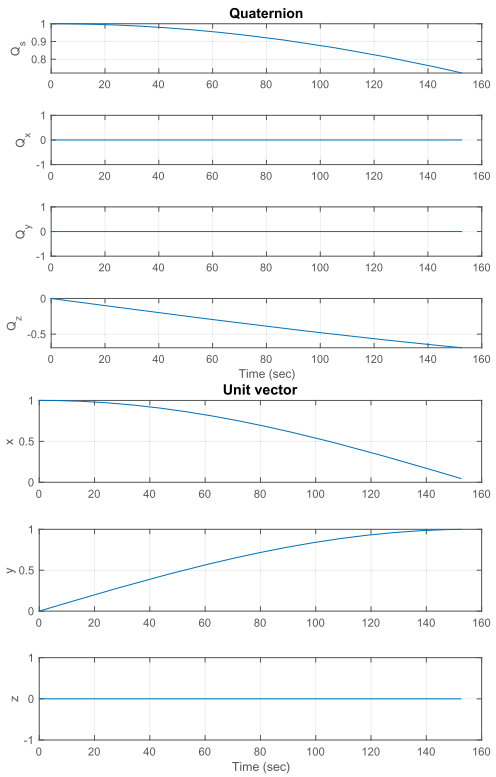
$$\dot{q} = \left(\frac{\omega}{2}\right) \begin{bmatrix} 0 & -a^T \\ a & -a^\times \end{bmatrix} q \tag{5.41}$$

Using the angular vector,

$$\dot{q} = \frac{1}{2} \begin{bmatrix} 0 & -\vec{\omega}^T \\ \vec{\omega} & -\vec{\omega}^\times \end{bmatrix} q \tag{5.42}$$

Example 5.1 shows an example of quaternion rotation. The rotation axis is chosen from the quaternion that rotates from the beginning frame to the final frame.

```
q1      = [1;0;0;0];
omega   = [0;0;0.01];
dT      = 0.1;
u2      = [0;1;0];

n       = 3000;
u       = zeros(3,n);
q(:,1)  = q1;
u(:,1)  = [1;0;0];
for k = 2:n
  q(:,k) = RK4(@RHSQuaternion,q(:,k-1),dT,0,omega
      );
  u(:,k) = QTForm(q(:,k),[1;0;0]);
       if( u(2,k) > 0.999)
    break
  end
end

q = q(:,1:k);
u = u(:,1:k);

[t,tL] = TimeLabl((0:k-1)*dT);
Plot2D(t,q,tL,{'Q_s' 'Q_x' 'Q_y' 'Q_z'},'
      Quaternion');
Plot2D(t,u,tL,{'x' 'y' 'z'},'Unit vector');

function qDot = RHSQuaternion(q,~,omega)

qDot = QIToBDot(q,omega);

end
```



Example 5.1: Quaternion rotation of a unit vector.

5.5.7 Small angles

Often, control systems deal with small deviations from a nominal attitude. In these cases, all attitude representations are the same and the attitude can be treated as a vector where

the order of the rotations does not matter. A quaternion derived from small angles is

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \approx \begin{bmatrix} 1 \\ \frac{\theta_1}{2} \\ \frac{\theta_2}{2} \\ \frac{\theta_3}{2} \end{bmatrix} / \sqrt{1 + \frac{\theta_1^2}{4} + \frac{\theta_2^2}{4} + \frac{\theta_3^2}{4}} \tag{5.43}$$

The term in the square root normalizes the equations. Since the angles are small this term can be omitted and the small-angle quaternion becomes

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \approx \begin{bmatrix} 1 \\ \frac{\theta_1}{2} \\ \frac{\theta_2}{2} \\ \frac{\theta_3}{2} \end{bmatrix} \tag{5.44}$$

5.5.8 Physical interpretation of the quaternion

Since the quaternion on the first appearance does not appear to impart a physical feeling for what attitude it represents, we will go through a few examples to show that with a little effort, a quaternion is as easy to visualize as Euler angles.

5.5.8.1 Single-axis rotation

The quaternion

$$\begin{bmatrix} 0.7071 \\ 0.7071 \\ 0.0 \\ 0.0 \end{bmatrix} \tag{5.45}$$

represents a pure rotation about the x-axis. The first argument is 0.7071 and equals $\cos(90°/2)$. We cannot tell the quadrant from the first argument. The second argument is the 1 component of the unit vector, (which in this case is

$$\begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix} \tag{5.46}$$

times the argument $\sin(90°/2)$). Since the sign is positive, the rotation must be a positive 90° rotation.

Large multiaxis rotations are somewhat more difficult to interpret.

5.5.8.2 Multiple-axis rotation

The quaternion

$$\begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \tag{5.47}$$

represents a rotation about an axis not aligned with x, y, or z. The first argument is 0.5 and equals the cos(120°/2). The sin(120°/2) is 0.866. Therefore the unit vector, a, is

$$\begin{bmatrix} 0.5773 \\ 0.5773 \\ 0.5773 \end{bmatrix} \tag{5.48}$$

This quaternion represents a +60° rotation about a or a -60° about $-a$. If the rotation is sufficiently small, the last three components of the quaternion represent half the equivalent angles.

5.5.8.3 Small rotation

The quaternion

$$\begin{bmatrix} 0.999 \\ 0.004 \\ 0.003 \\ -0.005 \end{bmatrix} \tag{5.49}$$

represents a small rotation. The first argument equals the cos(0.8°/2), which is small. Therefore the last three angles can be interpreted directly as angles (in radians). This quaternion represents approximately the rotation vector

$$\begin{bmatrix} 0.460° \\ 0.344° \\ 0.572° \end{bmatrix} \tag{5.50}$$

5.5.9 Incremental quaternion for maneuvers

Spacecraft pointing-control systems work on small angles from the nominal orientation. A PID or PD controller works quite well. The question arises as to how large reorientations are handled. Since the definition of a quaternion is a rotation axis and an angle, large-angle maneuvers can be implemented by finding the axis and angle that reorients the spacecraft and controlling about that axis. This reduces the three axis problem to a single axis problem.

The delta quaternion is

$$q_\delta = (q^T \otimes q_t)^T \tag{5.51}$$

where q is the current quaternion and q_t is the target. Define

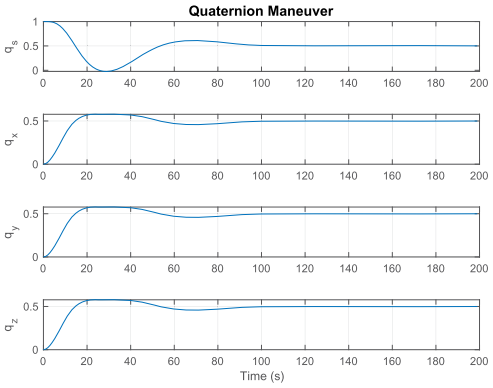
$$q_\delta = \begin{bmatrix} s \\ a \end{bmatrix} \tag{5.52}$$

The angle vector to the controller is

$$\theta = -2a \tag{5.53}$$

Example 5.2 shows an example of a quaternion maneuver.

```
n               = 200;
xP              = zeros(4,n);
dT              = 1;
xPID            = {[0;0] [0;0] [0;0]};

x               = [1;0;0;0;0;0;0];
qT              = [0.5;0.5;0.5;0.5];
accel           = zeros(3,1);

[a, b, c, d]    = PIDMIMO( 1, 0.7071, 0.1, 40,
       0.1, dT );

for k = 1:n
  q      = x(1:4);
  qTB    = QPose(QMult(QPose(q), qT));
  if(qTB(1) < 0 )
    qTB = -qTB;
  end
  theta = -2.0*qTB(2:4);

  for j = 1:3
    xPID{j}   =  a*xPID{j} + b*theta(j);
    accel(j)  = -c*xPID{j} - d*theta(j);
  end
  xP(:,k)         = q;
        x         = RK4(@RHS,x,dT,0,accel);
end

t  = (0:n-1)*dT;
yL = {'q_s' 'q_x' 'q_y' 'q_z' };
Plot2D(t,xP,'Time (s)',yL,'Quaternion Maneuver');

%% Right hand side [angle;rate]
function xDot = RHS(x,~,accel)

q       = x(1:4);
omega   = x(5:7);
qDot    = QIToBDot(q,omega);
xDot    = [qDot;accel];

end
```



Example 5.2: Quaternion maneuver. The spacecraft model is three double integrators.

5.5.10 Angle and unit vector to a quaternion

If you have an angle, θ, and a unit vector u you can create a quaternion

$$q = \begin{bmatrix} \cos\frac{\theta}{2} \\ u\sin\frac{\theta}{2} \end{bmatrix} \tag{5.54}$$

This is already normalized to one.

5.5.11 Axis-alignment quaternion

A frequent need is to align an axis in the spacecraft with an inertial axis. The Hubble Space Telescope does this whenever it begins an observation.

Take two unit vectors, u, and v. u is fixed to the spacecraft. v is in the inertial frame. First, find the dot product of the two vectors

$$d = u^T v \tag{5.55}$$

If $d = -1$ we have a special case. Otherwise, the scalar part of the quaternion is

$$s = \sqrt{2(1+d)} \tag{5.56}$$

The crossproduct of the vectors is the axis of rotation

$$c = u \times v \tag{5.57}$$

The final quaternion is

$$q = \begin{bmatrix} \frac{1}{2}s \\ \frac{c}{s} \end{bmatrix} \tag{5.58}$$

This should be normalized.

5.5.12 Small angles

Often satellites are planet pointing. This means that they rotate at orbit rate about one axis. The planet pointing frame is often local vertical/local horizontal (LVLH). If the angles are small, the transformation matrix from LVLH to the body is

$$.A = \begin{bmatrix} 1 & \theta_z & -\theta_y \\ -\theta_z & 1 & \theta_x \\ \theta_y & -\theta_x & 1 \end{bmatrix} \tag{5.59}$$

with θ_x about x, θ_y about y and θ_z about z. The angles and signs can be found by inspection by looking at Fig. 5.5.

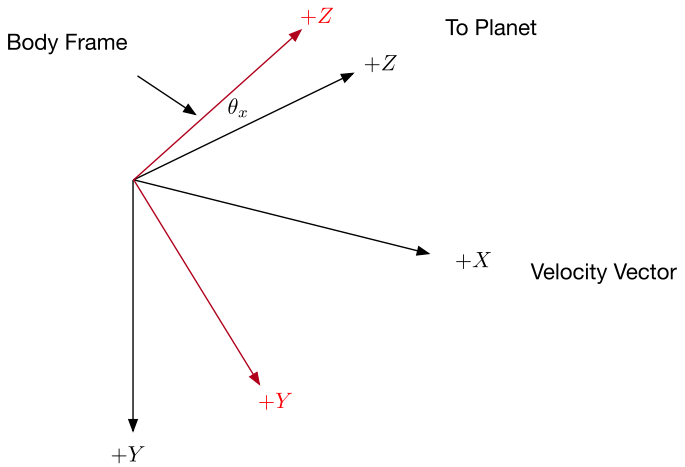


Figure 5.5 Coordinate transformations from LVLH to body with small angles. This diagram shows the roll angle, α. The body axes are in red.

Linearize the matrices so that $\cos\theta = 1$ and $\sin\theta = \theta$.

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \theta_x \\ 0 & -\theta_x & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\theta_y \\ 0 & 1 & 0 \\ \theta_y & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \theta_z & 0 \\ -\theta_z & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.60}$$

Multiplying the matrices and dropping products of angles

$$M = \begin{bmatrix} 1 & \theta_z & -\theta_y \\ -\theta_z & 1 & \theta_x \\ \theta_y & -\theta_x & 1 \end{bmatrix} \tag{5.61}$$

which is the same as Eq. (5.59).

5.5.13 Quaternion interpolation

Sometimes, it is desired to interpolate between two quaternions without doing quaternion propagation. The SLERP algorithm provides a way of interpolation. Assume there are two quaternions q_1 and q_2. Find their dot product

$$c = q_0^T q_1 \tag{5.62}$$

Let t be an array of the intermediate values for the quaternion. This ranges from 0 to 1. If c is less than zero, flip its sign and flip the sign of q_1. If c is far from 1 compute

$$\omega = \cos^{-1} c \tag{5.63}$$

$$s = \sin\omega \tag{5.64}$$

$$s_0 = \frac{\omega}{s}\sin(1-t) \tag{5.65}$$

$$s_1 = \frac{\sin(t\omega)}{s} \tag{5.66}$$

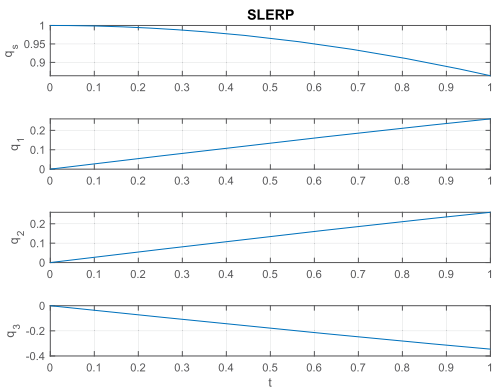
If c is near one, let $s_0 = 1 - t$ and $s_1 = t$. The interpolation is

$$q = s_0 q_0 + s_1 q_1 \tag{5.67}$$

This is often used in computer graphics.

Example 5.3 shows an example of quaternion rotation. The rotation axis is chosen from the quaternion that rotates from the beginning frame to the final frame.

```
q1 = QUnit([1;0.3;0.3;-0.4]);
q0 = [1;0;0;0];
t  = linspace(0,1);

q = QSLERP( q0, q1, t );

yL = {'q_s' 'q_1' 'q_2' 'q_3'};

Plot2D(t,q,'t',yL,'SLERP')
```



Example 5.3: Quaternion interpolation.

CHAPTER 6

Attitude dynamics

6.1. Space story

We were flying a spacecraft in transfer orbit that had a liquid apogee engine (LAE). This meant that unlike our other spacecraft it had an oxidizer tank. Our particular LAE engines used hydrazine, so the same propellant could be used in the monopropellant engines and the LAE. We were spinning at 10 rpm for our first LAE maneuver. When it started up we noticed a slight oscillation that damped quickly. Nutation was damped by the nutation controllers that had been designed to correct nutation at tip-off when the tanks were full. As speed was important at the tip-off it did not have any roll-off. This made sense because there were no frequencies greater than nutation when the tanks were full.

On the next burn, we lowered the spin rate to reduce fuel consumption. We started the burn and the control system started limit-cycling. Control was stable so I did not stop the burn. We had an extra year of fuel so a little wasted during an LAE burn was not important. As the customer wanted to get to mission orbit fast, this was a reasonable decision. Management did not agree but the mission went on. The problem was that as the tanks emptied slosh motion of the fuel was greater and it was worse at the lower frequency. We spun up to 10 rpm and the problem went away.

6.2. Introduction

This chapter provides an introduction to spacecraft-attitude dynamics. In this chapter, we will discuss rigid-body rotation. This motion is in general, nonlinear and all three axes are coupled. The following sections cover multibody dynamics. The models presented are central-hub models. We first discuss the gyrostat that is a rigid body with any number of symmetric rotors. The gyrostat is important for spacecraft with reaction wheels. We then cover models that allow for more than one degree of rotational freedom at the joint and include the effects of center-of-mass movement. Finally, we discuss a spacecraft in which part of the spacecraft can undergo infinitesimal displacements. This is representative of a spacecraft with an attached flexible structure.

The approach taken in most cases in this chapter is to derive the equations of motion based on momentum conservation. The planar slosh problem uses Lagrange's method. Another well-known method is Kane's method that involves generalized accelerations [1]. Many general-purpose computer codes exist for modeling complex multibody systems, such as topological trees with and without closed loops. Some software packages use symbolic derivation to produce compact and efficient multibody models.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00017-6

6.3. Inertia matrix

6.3.1 Definition

The inertia matrix represents the resistance to the rotation of a spacecraft. The inertia matrix is positive-definite-symmetric, which means that the direction of rotation does not matter. The nine elements are

$$I_{xx} = \int\int\int_V \rho(x, y, z)(y^2 + z^2)\ \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{6.1}$$

$$I_{yy} = \int\int\int_V \rho(x, y, z)(x^2 + z^2)\ \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{6.2}$$

$$I_{zz} = \int\int\int_V \rho(x, y, z)(x^2 + y^2)\ \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{6.3}$$

$$I_{xy} = -\int\int\int_V \rho(x, y, z)xy\ \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{6.4}$$

$$I_{xz} = -\int\int\int_V \rho(x, y, z)xz\ \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{6.5}$$

$$I_{yz} = -\int\int\int_V \rho(x, y, z)yz\ \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{6.6}$$

where the vectors are from the center-of-mass. Positive-definite means that

$$z^T Iz > 0 \tag{6.7}$$

for all nonzero vectors z with real entries. All inertia matrices can be transformed, through the appropriate change in axes, into a diagonal matrix.

The center-of-mass is found from

$$c_x = \frac{1}{M}\int\int\int_V \rho(x, y, z)x\ \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{6.8}$$

$$c_y = \frac{1}{M}\int\int\int_V \rho(x, y, z)y\ \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{6.9}$$

$$c_z = \frac{1}{M}\int\int\int_V \rho(x, y, z)z\ \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{6.10}$$

where c is the vector to the center-of-mass and

$$M = \int\int\int_V \rho(x, y, z)\ \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{6.11}$$

6.3.2 Inertia matrix from components

The diagonal terms must all be positive. $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$, and $I_{yz} = I_{zy}$. It is often useful to decompose a spacecraft into a set of point masses. The inertia matrix for N

point masses is

$$I = -\sum_{k=1}^{N} m_k r_k^{\times} r_k^{\times} \tag{6.12}$$

where

$$r_k^{\times} = \begin{bmatrix} 0 & -z_k & y_k \\ z_k & 0 & -x_k \\ -y_k & x_k & 0 \end{bmatrix} \tag{6.13}$$

and the contribution of one point mass is

$$I_k^{\times} = m_k \begin{bmatrix} y_k^2 + z_k^2 & -x_k y_k & -x_k z_k \\ -x_k y_k & x_k^2 + z_k^2 & -y_k z_k \\ -x_k z_k & -y_k z_k & x_k^2 + y_k^2 \end{bmatrix} \tag{6.14}$$

For point masses

$$c = \frac{\sum_{k=1}^{N} m_k r_k}{\sum_{k=1}^{N} m_k} \tag{6.15}$$

The point-mass approach is very useful when individual inertias for components are not available. This method can be expanded to include rigid bodies instead of point masses,

$$I = \sum_{k=1}^{N} \left(B_k I_k B_k^T - m_k r_k^{\times} r_k^{\times} \right) \tag{6.16}$$

where B_k transforms from the frame in which the inertia matrix is calculated to the spacecraft body frame.

6.3.3 Common inertia matrices

Table 6.1 gives inertia matrices for some common geometric solids. m is the mass, r is the radius for the sphere and cylinder, and l is the height for a cylinder. x, y, and z are box dimensions. The cylinder is aligned with the z-axis. Example 6.1 computes the inertia of a satellite composed of three boxes. Two represent the solar panels and one the bus. This is a reasonable first cut at the inertia of the satellite. The deployment struts are ignored because they have low masses. The satellite bus is treated as a solid since it is full of all the bus components. The model is shown in Fig. 6.1.

Table 6.1 Common inertia matrices.

Type	Inertia matrix
Solid Box	$\frac{m}{12}\begin{bmatrix} y^2+z^2 & 0 & 0 \\ 0 & x^2+z^2 & 0 \\ 0 & 0 & x^2+y^2 \end{bmatrix}$
Solid cylinder	$\frac{m}{12}\begin{bmatrix} 3r^2+l^2 & 0 & 0 \\ 0 & 3r^2+l^2 & 0 \\ 0 & 0 & 6r^2 \end{bmatrix}$
Solid sphere	$\frac{2mr^2}{5}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

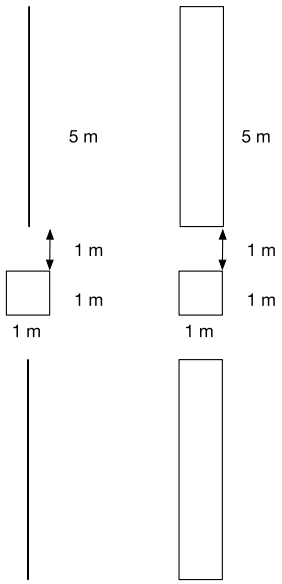


Figure 6.1 A satellite represented by three boxes.

6.4. Rigid body

The dynamical equations for a spacecraft can be derived by first writing the total angular momentum for the system in the inertial frame.

$$H = Ah \tag{6.17}$$

where h is the angular momentum in the body frame and the transformation matrix A transforms from the body-fixed coordinate frame to the inertial reference frame. The

```
%% Inertia calculation

% Solar panel
x = 1;
y = 5;
z = 0.01;
mSP = 2;
rN = [0;y/2+x;0];
rS = [0;-y/2-x;0];
iSP = (mSP/12)*diag([y^2+z^2 x^2+z^2 x^2+y^2])

% Bus
x = 1;
y = 1;
z = 1;
mBus = 10;
iBus = (mBus/12)*diag([y^2+z^2 x^2+z^2 x^2+y^2])

iTotal = iBus + 2*iSP - mSP*(SkewSq(rN) + SkewSq(
    rS))
```

```
iSP =

    4.1667         0         0
         0    0.1667         0
         0         0    4.3333


iBus =

    1.6667         0         0
         0    1.6667         0
         0         0    1.6667


iTotal =

   59.0000         0         0
         0    2.0000         0
         0         0   59.3333
```

Example 6.1: Inertia matrix for a satellite represented by three boxes.

torque on the spacecraft is

$$\begin{aligned} AT &= \dot{H} = \dot{A}h + A\dot{h} \\ \dot{A} &= A\omega^{\times} \\ T &= \dot{h} + \omega^{\times}h \end{aligned} \tag{6.18}$$

where T is defined in the body frame and ω is the angular rate of the body with respect to the inertial frame measured in the body frame. For a rigid body $h = I\omega$ so

$$T = I\dot{\omega} + \omega^{\times}I\omega \tag{6.19}$$

which is Euler's equation. I does not have a time derivative when the spacecraft is rigid. The inertia matrix, I, is positive-definite-symmetric. This fact is important when we solve for the angular acceleration by solving the set of linear equations

$$I\dot{\omega} = T - \omega^{\times}I\omega \tag{6.20}$$

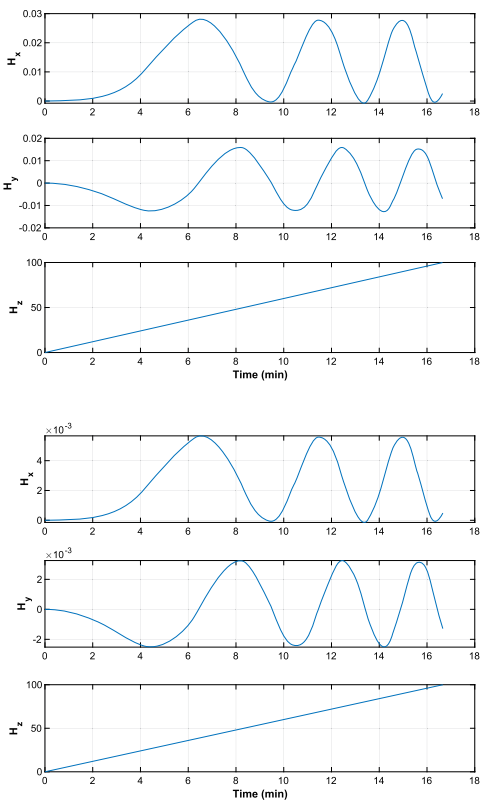
These equations are written about the spacecraft's center-of-mass. This decouples the angular motion from the translational motion.

Generally, we do not compute the inverse of I, rather we solve the set of linear equations for $\dot{\omega}$. As I is positive-definite-symmetric, we can solve this set of equations using a linear equation solver without pivoting, thus saving much computation. For a 3-by-3 system with a constant inertia matrix, we can precompute the inverse and not go through the process of using a solver. Pivoting is the process by which we find an element in a column with the largest absolute value and then do row shifts so that this number is in the row being considered.

A is a transformation matrix that we normally represent as a quaternion. Quaternions are discussed in Section 5.5. To demonstrate rigid-body dynamics define an inertially fixed torque about the z-axis and a fully populated inertia matrix as in Example 6.2.

The first four elements of the state vector, x, is the quaternion that transforms from the inertial frame to the body frame with a starting value of [1;0;0;0]. The last three elements are the angular rate in the body frame that is [0;0;0]. The equations of motion are in the function FRB and the function RK4 is the fourth-order Runge–Kutta integrator. QTForm uses the quaternion to transform from the body frame to the inertial frame. QForm transforms from the inertial frame to the body frame.

```
inertia     = [2000 300  -200;...
               300 4000  -100;...
              -200 -100 1000];
tEnd        = 1000;
dT          = 1.0;
n         = tEnd/dT;
torque      = [0;0;0.1];
xP           = zeros(3,n);
x           = [1;0;0;0;0;0;0];

for k = 1:n
  xP(:,k) = QTForm(x(1:4),inertia*x(5:7));
  tBody    =  QForm( x(1:4), torque );
  x = RK4( @RHS, x, dT, 0, inertia, tBody );
end

yL = {'H_x' 'H_y' 'H_z'};
s = sprintf('Rigid Body dT = %1.1f s',dT);
TimeHistory( (0:(n-1))*dT, xP, yL , s)

dT   = 0.2;
n = tEnd/dT;
xP   = zeros(3,n);
x    = [1;0;0;0;0;0;0];

for k = 1:n
  xP(:,k) = QTForm(x(1:4),inertia*x(5:7));
  tBody    =  QForm( x(1:4), torque );
  x = RK4( @RHS, x, dT, 0, inertia, tBody );
end

s = sprintf('Rigid Body dT = %1.1f s',dT);
TimeHistory( (0:(n-1))*dT, xP, yL, s)

%% Right-hand-side for a rigid body
function xDot = RHS( x, ~, inertia, torque )

q          = x(1:4);
omega      = x(5:7);
qDot       = QIToBDot(q,omega);
omegaDot   = inertia\...
             (torque-Cross(omega,inertia*omega));

xDot       = [qDot;omegaDot];

end
```



Example 6.2: Rigid-body dynamics showing momentum conservation. Note the Y-axis scale. The top simulation has a time step of 1 second, the bottom has a time step of 0.2 seconds.

The top plot shows the simulation with a time step of 0.2 s. The z-axis inertial momentum grows linearly, as expected. However, an oscillation is seen in the x- and y-axes inertial momentum. This cannot happen physically and is an artifact of numerical integration. This can be confirmed by running the same simulation with a time step of 1 s, as shown in the lower plot.

The oscillation magnitude in the lower plot is much larger than in the first plot. Whenever you do this kind of simulation it is always important to verify that angular momentum is conserved. When an integrator that does not have error correction is used, such as RK4, you should change the time step and observe the inertial angular momentum. If you use an error-correcting integrator, such as RK45 or MATLAB®'s ode113, you can change the error tolerance and observe the result. If tightening the error tolerance does not improve the result, there is an error in the dynamics.

6.5. Gyrostat

A gyrostat is a spacecraft that is rigid except for symmetric rotors that spin about axes fixed to the body frame. Many spacecraft are gyrostats since reaction wheels can control the spacecraft's attitude without the need to expend propellant. A gyrostat is used to represent spacecraft with reaction wheels. The momentum for a gyrostat is

$$H = A\left(I\omega + \sum_k u_k J_k(\Omega_k + u_k^T\omega)\right) \tag{6.21}$$

where Ω_k is the angular rate of the kth wheel about its spin axis, ω are the body rates, J_k is the spin-axis moment of inertia, u_k is the unit vector of the wheel spin axis measured in the body frame, and I is the inertia of the rigid body excluding the wheels. The equations of motion are then

$$T = I\dot{\omega} + \omega^{\times}\left(I\omega + \sum_k u_k J_k(\Omega_k + u_k^T\omega)\right) + \sum_k u_k J_k(\dot{\Omega}_k + u_k^T\dot{\omega}) \tag{6.22}$$

$$T_k = J_k(\dot{\Omega}_k + u_k^T\dot{\omega}) \tag{6.23}$$

The transverse axes' inertias are lumped into the core body inertia. T is the body-fixed torque on the spacecraft and T_k is the scalar torque on each wheel. These equations give $3 + n$ equations of motion. To integrate these equations you need first to solve for the angular acceleration of the core, and then substitute the angular-acceleration vector into the reaction-wheel equations.

$$\dot{\omega} = I^{-1}\left(T - \omega^{\times}\left(I\omega + \sum_k u_k J_k(\Omega_k + u_k^T\omega)\right) - \sum_k u_k T_k\right) \tag{6.24}$$

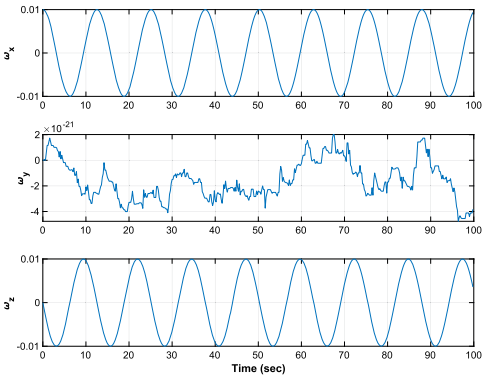
$$\dot{\Omega}_k = \frac{T_k}{J_k} - u_k^T \dot{\omega} \tag{6.25}$$

Spacecraft with reaction and momentum wheels are the most popular type of spacecraft design today. Some examples are

- Boeing HS702 communications Satellite;
- Lockheed Martin A2100 communications Satellite;
- Lockheed Martin GPS IIR navigation satellite;
- NASA/JPL Cassini scientific satellite.

Example 6.3 shows the effect of having a bias momentum on a spacecraft. The spacecraft has a 1-N m^2 wheel aligned with the y-axis spinning at 100 rad/s. One of the transverse rates (about the x-axis) is set to 0.01 rad/s. The spacecraft has a diagonal inertia matrix that eliminates all Euler coupling between the axes. The momentum bias introduces coupling that causes the angular momentum to be transferred between the x- and z-axes thus maintaining the overall pointing orientation of the spacecraft. The oscillation is called *nutation*. The y-axis rate in the plot is due to a numerical integration error.

```
dRHS              = RHSGyrostat;
dRHS.uWheel       = [0;1;0];
dRHS.inrWheel     = 1;
dRHS.inr          = 200*eye(3);
dRHS.torqueWheel  = 0;
dRHS.friction     = struct('coulomb',0,'damping'
      ,0,'kCoulomb',0);
tEnd              = 100;
dT                = 0.2;
nSim              = floor(tEnd/dT);
xP                = zeros(3,nSim);
x                 = [[1;0;0;0];[0.01;0;0];100];
for k = 1:nSim
  xP(:,k) = x(5:7);
  x       = RK4(@RHSGyrostat,x,dT,0,dRHS);
end
yL = {'\omega_x' '\omega_y' '\omega_z'};
TimeHistory( (0:(nSim-1))*dT,xP,yL,'Momentum Bias
      ')
```



Example 6.3: Momentum-bias spacecraft nutation.

6.6. Dual spin

A special case of a gyrostat is a dual-spin spacecraft. This has one spun part and one part rotating at orbit rate. The dynamical equations are

$$T = I\dot{\omega} + \omega^{\times}\left(I\omega + uJ(\Omega + u^T\omega)\right) + uJ(\dot{\Omega} + u^T\dot{\omega}) \tag{6.26}$$

$$T_w = J(\dot{\Omega} + u^T\dot{\omega}) \tag{6.27}$$

where

$$u = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{6.28}$$

Assume that the body rates are

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y + \omega_o \\ \omega_z \end{bmatrix} \tag{6.29}$$

where the body rates are small, ω_o is the orbit rate, and y is along the orbit normal. Assume that the inertia matrix is diagonal and that Ω is constant. We then get

$$T = \left(I + uJu^T\right)\dot{\omega} + \omega^{\times}\left(I\omega + uJ(\Omega + u^T\omega)\right) \tag{6.30}$$

The wheel torque compensates for $\dot{\omega}_y$. Note that

$$uJu^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & J & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6.31}$$

The linearized equations are

$$T_x = I_x\dot{\omega}_x - \omega_z\left(\omega_o(I_y - I_z) + \Omega J\right) \tag{6.32}$$

$$T_y = I_y\dot{\omega}_y \tag{6.33}$$

$$T_x = I_z\dot{\omega}_z + \omega_x\left(\omega_o(I_y - I_x) + \Omega J\right) \tag{6.34}$$

where I is now the total spacecraft inertia, including the wheel.

$$\begin{bmatrix} \omega_x \\ \omega_z \end{bmatrix} = \frac{\begin{bmatrix} s & \omega_o(I_y - I_z) + \Omega J \\ -\omega_o(I_y - I_x) - \Omega J & s \end{bmatrix}}{s^2 + \left(\omega_o(I_y - I_z) + \Omega J\right)\left(\omega_o(I_y - I_x) + \Omega J\right)} \begin{bmatrix} a_x \\ a_z \end{bmatrix} \tag{6.35}$$

The nutational modes are pure oscillatory as long as these two conditions are met

$$\Omega J > \omega_o(I_y - I_x) \tag{6.36}$$

$$\Omega J > \omega_o(I_y - I_z) \tag{6.37}$$

6.7. Gravity gradient

An Earth-pointing spacecraft, with the y-axis along the negative orbit normal (the LVLH frame), is modeled by the following dynamical equation

$$I\dot{\omega} + \omega^{\times} I\omega = T_{gg} + T \tag{6.38}$$

The coordinate frame, which is rotating at a steady rotation rate n about the $-y$-axis is shown in Fig. 6.2.

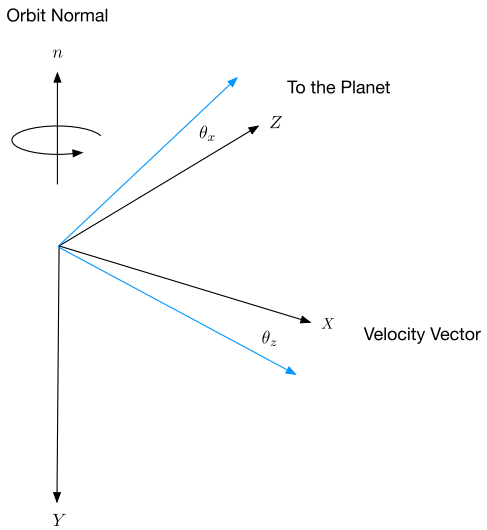


Figure 6.2 Rotating coordinate frame.

This leads to the following kinematical equations by inspection

$$\omega = \begin{bmatrix} \dot{\theta}_x - n\theta_z \\ \dot{\theta}_y - n \\ \dot{\theta}_z + n\theta_x \end{bmatrix} \tag{6.39}$$

and

$$\dot{\omega} = \begin{bmatrix} \ddot{\theta}_x - n\dot{\theta}_z \\ \ddot{\theta}_y \\ \ddot{\theta}_z + n\dot{\theta}_x \end{bmatrix} \tag{6.40}$$

where n is the orbit rate and where the angles are all small angles from the rotating frame. We will first do the traditional derivation, and then do one that includes the off-diagonal inertia terms.

The gravity-gradient disturbance torque arises from the inertia matrix. The vector torque is

$$T = 3\mu \frac{r^{\times} Ir}{|r|^5} \tag{6.41}$$

where r is the vector to the satellite from the center of the Earth, r is along the $-z$-axis of the rotating frame. As long as μ and r are in consistent units, the distance units that are used drop out and the result is $1/s^2$. The torque will then be in the units determined by the units of the inertia matrix. Substituting, we get

$$T = 3n^2 u^{\times} Iu \tag{6.42}$$

where the unit vector

$$u = \frac{r}{|r|} \tag{6.43}$$

is approximately

$$u = \begin{bmatrix} -\theta_y \\ \theta_x \\ 1 \end{bmatrix} \tag{6.44}$$

The linearized gravity-gradient torque is

$$T_{gg} = 3n^2 \begin{bmatrix} (I_z - I_y)\theta_x \\ (I_z - I_x)\theta_y \\ 0 \end{bmatrix} \tag{6.45}$$

and n is the orbit rate. Assume that the inertia matrix is diagonal and use the linearized body rates. Euler's equations become

$$T_x = I_x \ddot{\theta}_x + n\dot{\theta}_z(I_y - I_z - I_x) - n^2\theta_x(I_z - I_y) \tag{6.46}$$

$$T_y = I_y \ddot{\theta}_y \tag{6.47}$$

$$T_z = I_z \ddot{\theta}_z + n\dot{\theta}_x(I_z + I_x - I_y) + n^2\theta_z(I_y - I_x) \tag{6.48}$$

Adding the gravity-gradient torque we get three second-order differential equations

$$T_x = I_x \ddot{\theta}_x + n\dot{\theta}_z(I_y - I_z - I_x) - 4n^2\theta_x(I_z - I_y) \tag{6.49}$$

$$T_y = I_y \ddot{\theta}_y + 3n^2(I_x - I_z)\theta_y \tag{6.50}$$

$$T_z = I_z \ddot{\theta}_z + n\dot{\theta}_x(I_z + I_x - I_y) + n^2\theta_z(I_y - I_x) \tag{6.51}$$

All of the stiffness terms, that is terms that include angle, must have positive coefficients for stability. The x and z equations are coupled. The y equation is a harmonic oscillator.

Let

$$k_x = 4n^2 \frac{I_y - I_z}{I_x} \tag{6.52}$$

$$k_y = 3n^2 \frac{I_x - I_z}{I_y} \tag{6.53}$$

$$k_z = n^2 \frac{I_y - I_x}{I_z} \tag{6.54}$$

$$c_x = n \frac{I_y - I_z - I_x}{I_x} \tag{6.55}$$

$$c_z = n \frac{I_z + I_x - I_y}{I_z} \tag{6.56}$$

Since y is decoupled, $I_x > I_z$ is required. The frequency of oscillation for y is

$$\Omega_y = n\sqrt{3k_y} \tag{6.57}$$

The xz frequencies are the libration frequencies and are found in the following equations

$$a_x = \ddot{\theta}_x + c_x\dot{\theta}_z + k_x\theta_x \tag{6.58}$$

$$a_z = \ddot{\theta}_z + c_z\dot{\theta}_x + k_z\theta_z \tag{6.59}$$

The full 3-axis system in state-space form is

$$\begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \\ \ddot{\theta}_x \\ \ddot{\theta}_y \\ \ddot{\theta}_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -k_x & 0 & 0 & 0 & 0 & -c_x \\ 0 & -k_y & 0 & 0 & 0 & 0 \\ 0 & 0 & -k_z & -c_z & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \\ \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{bmatrix} \tag{6.60}$$

Write the equations for x and z in the s domain

$$\begin{bmatrix} s^2 + k_x & c_x s \\ c_z s & s^2 + k_z \end{bmatrix} \begin{bmatrix} \theta_x \\ \theta_z \end{bmatrix} = \begin{bmatrix} a_x \\ a_z \end{bmatrix} \tag{6.61}$$

where a_x and a_z are the angular accelerations in x and z. The denominator will be quadratic in s^2 and the analytical solution is straightforward.

$$\begin{bmatrix} \theta_x \\ \theta_z \end{bmatrix} = \frac{\begin{bmatrix} s^2 + k_z & -c_x s \\ -c_z s & s^2 + k_x \end{bmatrix}}{s^4 + (k_x + k_z - c_x c_z)s^2 + k_x k_z} \begin{bmatrix} a_x \\ a_z \end{bmatrix} \tag{6.62}$$

Write the denominator as

$$s^4 + b_1 s^2 + b_0 \tag{6.63}$$

The eigenvalues are

$$s^2 = -\frac{b_1}{2} \pm \sqrt{\frac{b_1^2}{4} - b_0^2} \tag{6.64}$$

For s to have pure imaginary roots, the roots for s^2 must be negative. This requires

$$b_0 > 0 \tag{6.65}$$

$$b_1 > 0 \tag{6.66}$$

$$b_1 > 2b_0 \tag{6.67}$$

The last condition insures that the term in the square root is positive. Of course, if this condition is true then the second is also true. If these criteria are met, the system is an undamped oscillator. The first requires that

$$I_x > I_y > I_z \tag{6.68}$$

To provide damping we need to add

$$a_x = -2\zeta\sqrt{k_x}\dot{\theta}_x \tag{6.69}$$

Generally, the problem is we have bus inertia, I_0 to which we want to add a boom to give us gravity-gradient stability. Define

$$\dot{\omega} = \ddot{\theta} + D\dot{\theta} \tag{6.70}$$

$$\omega = \dot{\theta} + D\theta + N \tag{6.71}$$

where

$$D = \begin{bmatrix} 0 & 0 & -n \\ 0 & 0 & 0 \\ n & 0 & 0 \end{bmatrix} \tag{6.72}$$

$$N = \begin{bmatrix} 0 \\ -n \\ 0 \end{bmatrix} \tag{6.73}$$

The linearized Euler's equation with gravity gradient becomes

$$T = I\ddot{\theta} + ID\dot{\theta} + (\dot{\theta} + D\theta + N)^{\times} I(\dot{\theta} + D\theta + N) - G\theta \tag{6.74}$$

Removing products of small quantities

$$T = I\ddot{\theta} + ID\dot{\theta} + \dot{\theta}^{\times} IN + (D\theta)^{\times} IN + N^{\times} I\dot{\theta} + N^{\times} ID\theta - G\theta \tag{6.75}$$

We use the matrix identities

$$(D\theta)^{\times} = D\theta^{\times} D^{T} \tag{6.76}$$

$$a^{\times} b = -b^{\times} a \tag{6.77}$$

and

$$G = 3n^2 \begin{bmatrix} I_z - I_y & I_{xy} & 0 \\ I_{xy} & I_z - I_x & 0 \\ -I_{xz} & I_{yz} & 0 \end{bmatrix} \tag{6.78}$$

results in

$$T = I\ddot{\theta} + ID\dot{\theta} - (IN)^{\times}\dot{\theta} - (D^{T} IN)^{\times} D\theta + N^{\times} I\dot{\theta} + N^{\times} ID\theta - G\theta \tag{6.79}$$

We can then put this into state-space form to compute the eigenvalues.

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & E \\ -I^{-1}(N^{\times} ID - G - (IN)^{\times} D) & -I^{-1}(ID - (IN)^{\times} + N^{\times} I) \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \tag{6.80}$$

where E is a 3-by-3 identity matrix. Example 6.4 compares the two formulations.

Higher-order models for gravity gradient are also available [2]. The following includes only the J_2 term.

$$T_g = \frac{3\mu}{|r|^3} u_r^{\times} I u_r + \frac{\mu J_2 R_p^2}{2|r|^5} \left[30 u_r^T u_n (u_r^{\times} I u_n + u_n^{\times} I u_r) + (15 - 105(u_r^T u_n)^2) u_n^{\times} I u_r + 6 u_n^{\times} I u_n \right] \tag{6.81}$$

where J_2 is the gravitational harmonic, u_r is the unit vector from the spacecraft center-of-mass to the center of the planet, R_p is the planet radius, and u_n is the unit vector along the rotation axis of the planet. Since J_2 is small, this term is very small, particularly when $|r| >> R_p$.

6.8. Nutation dynamics

A spinning spacecraft is modeled by Euler's equation

$$I\dot{\omega} + \omega^{\times} I\omega = T \tag{6.82}$$

```
%% Gravity Gradient

inr = [3 0 0;0 2 0;0 0 1];

n   = 2*pi/(90*60);

nSq = n^2;

kX  = 4*nSq*(inr(2,2) - inr(3,3))/inr(1,1);
kY  = 3*nSq*(inr(1,1) - inr(3,3))/inr(2,2);
kZ  =   nSq*(inr(2,2) - inr(1,1))/inr(3,3);
cX  = n*(inr(2,2) - inr(3,3) - inr(1,1))/inr(1,1)
      ;
cZ  = n*(inr(3,3) + inr(1,1) - inr(2,2))/inr(3,3)
      ;

a   = [zeros(3,3) eye(3);-diag([kX kY kZ])...
  [0 0 -cX;0 0 0;-cZ 0 0]];

disp('Eigenvalues for the matrix')
eig(a)

disp('Roots for roll/yaw')
roots([1 0 (kX+kZ - cX*cZ) 0 kX*kZ])

theta = [0.001;0.002;0.003];
thetaDot = theta/100;

N = [0;-n;0];
D = [0 0 -n;0 0 0;n 0 0];
G = 3*n^2*[inr(3,3)-inr(2,2) inr(1,2) 0;...
 inr(1,2) inr(3,3)-inr(1,1) 0;...
 -inr(1,3) inr(2,3) 0];


a = [zeros(3,3) eye(3);-inr\(Skew(N)*inr*D - G -
      Skew(inr*N)*D)...
-inr\(inr*D - Skew(inr*N) + Skew(N)*inr)];

disp('Eigenvalues for the full matrix')
eig(a)
```

```
Eigenvalues for the matrix

ans =

  -8.9425e-04 + 0.0000e+00i
   8.9425e-04 + 0.0000e+00i
   5.4210e-20 + 1.7482e-03i
   5.4210e-20 - 1.7482e-03i
   0.0000e+00 + 2.0153e-03i
   0.0000e+00 - 2.0153e-03i

Roots for roll/yaw

ans =

   3.3881e-19 + 1.7482e-03i
   3.3881e-19 - 1.7482e-03i
  -8.9425e-04 + 0.0000e+00i
   8.9425e-04 + 0.0000e+00i

Eigenvalues for the full matrix

ans =

   8.9425e-04 + 0.0000e+00i
  -8.9425e-04 + 0.0000e+00i
  -2.8309e-101 + 1.7482e-03i
  -2.8309e-101 - 1.7482e-03i
   0.0000e+00 + 2.0153e-03i
   0.0000e+00 - 2.0153e-03i
```

Example 6.4: Two formulations for gravity gradient. The second can accommodate off-diagonal terms.

where T is torque, I is inertia, and ω is the body angular rate expressed in the body frame. Assume that the spacecraft inertia matrix is diagonal. The equations become

$$I_x\dot{\omega}_x + \omega_y\omega_z(I_z - I_y) = T_x \tag{6.83}$$

$$I_y\dot{\omega}_y + \omega_x\omega_z(I_x - I_z) = T_y \tag{6.84}$$

$$I_z\dot{\omega}_z - \omega_x\omega_y(I_x - I_y) = T_z \tag{6.85}$$

Define

$$k_x = \frac{I_z - I_y}{I_x} \tag{6.86}$$

$$k_y = \frac{I_x - I_z}{I_y} \tag{6.87}$$

$$k_z = \frac{I_x - I_y}{I_z} \tag{6.88}$$

and

$$a_x = \frac{T_x}{I_x} \tag{6.89}$$

$$a_y = \frac{T_y}{I_y} \tag{6.90}$$

$$a_z = \frac{T_z}{I_z} \tag{6.91}$$

we then get

$$\dot{\omega}_x + \omega_y \omega_z k_x = a_x \tag{6.92}$$

$$\dot{\omega}_y + \omega_x \omega_z k_y = a_y \tag{6.93}$$

$$\dot{\omega}_z - \omega_x \omega_y k_z = a_z \tag{6.94}$$

Assume also that the angular-momentum vector is the sum of a set of small angular rates and a large spin rate about the y-axis.

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \begin{bmatrix} 0 \\ \Omega \\ 0 \end{bmatrix} \tag{6.95}$$

where ω is now a vector of small angular rates and Ω is constant. We then get linearized dynamical equations

$$\dot{\omega}_x + k_x \Omega \omega_z = a_x \tag{6.96}$$

$$\dot{\omega}_y = a_y \tag{6.97}$$

$$\dot{\omega}_z - k_z \Omega \omega_x = a_z \tag{6.98}$$

where $k = k_x \Omega$. The nutational equations are

$$\begin{bmatrix} s & k_x \Omega \\ -k_z \Omega & s \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_z \end{bmatrix} = \begin{bmatrix} a_x \\ a_z \end{bmatrix} \tag{6.99}$$

solving

$$\begin{bmatrix} \omega_x \\ \omega_z \end{bmatrix} = \frac{\begin{bmatrix} s & -k_x \Omega \\ k_z \Omega & s \end{bmatrix}}{s^2 + k_x k_z \Omega^2} \begin{bmatrix} a_x \\ a_z \end{bmatrix} \tag{6.100}$$

the nutation frequency is

$$\omega_n = \Omega \sqrt{k_x k_z} \tag{6.101}$$

A major-axis spinner is

$$I_y > I_x > I_z \tag{6.102}$$

A minor-axis spinner

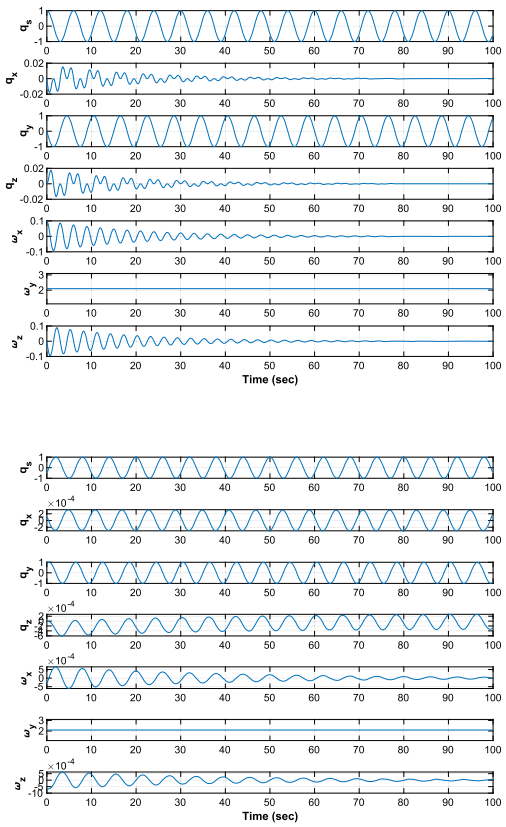
$$I_x > I_z > I_y \tag{6.103}$$

An intermediate-axis spinner is, for example,

$$I_x > I_y > I_z \tag{6.104}$$

For a major- or minor-axis spinner, $k_x k_z > 0$, and the spacecraft spin is stable. For an intermediate-axis spinner, $k_x k_z < 0$, and therefore it is unstable. The two cases are shown in Example 6.5. The attitude motion does not damp with the minor-axis spinner.

```
%% Nutation

clear d

% MajorAxis
d.inr    = diag([1 2 1]); % Major axis
d.damp   = 0.1;
x        = [1;0;0;0;0.1;20*pi/30;0];
n        = 1000;
xP       = zeros(7,n);
dT       = 0.1;
for k = 1:n
  xP(:,k) = x;
  x       = RK4(@RHS,x,dT,0,d);
end

t = (0:n-1)*dT;
yL = {'q_s' 'q_x' 'q_y' 'q_z'...
  '\omega_x' '\omega_y' '\omega_z' };
TimeHistory(t,xP,yL,'Major Axis');

% Minor raxis
d.inr    = diag([2 1 2]); % Major axis

for k = 1:n
  xP(:,k) = x;
  x       = RK4(@RHS,x,dT,0,d);
end

TimeHistory(t,xP,yL,'Minor Axis');


function xDot = RHS(x,~,d )

q = x(1:4);
omega = x(5:7);
t = -d.damp*[x(5);0;0];

omegaDot = d.inr\(t - Cross(omega,d.inr*omega));

xDot = [QIToBDot(q,omega);omegaDot];

end
```



Example 6.5: Major- and minor-axis spin.

6.9. Planar slosh model

Fig. 6.3 shows the planar slosh model. It consists of a slosh mass and a central body with inertia and mass.

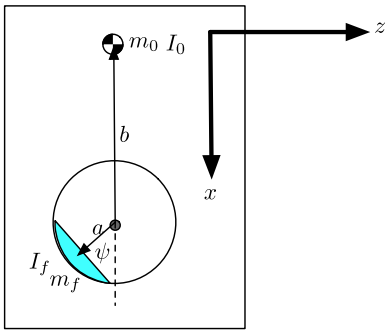


Figure 6.3 The planar slosh system.

The equations of motion are derived using the Lagrangian, which is the difference between the kinetic and potential energies.

$$L = T - V \tag{6.105}$$

The equations of motion are found from

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = 0 \tag{6.106}$$

where q is an independent coordinate.

The position of the center-of-mass with respect to the fuel tank center-of-mass is

$$\vec{r} = (x - b)\hat{i} + z\hat{k} \tag{6.107}$$

The velocity of the point is

$$\vec{v} = \dot{\vec{r}} - \omega^{\times}\vec{r} \tag{6.108}$$

where

$$\omega^{\times} = \begin{bmatrix} 0 & 0 & -\dot{\theta} \\ 0 & 0 & 0 \\ \dot{\theta} & 0 & 0 \end{bmatrix} \tag{6.109}$$

which is

$$\vec{v} = \left(\dot{x} + \dot{\theta}z\right)\hat{i} + \left(\dot{z} + \dot{\theta}(b - x)\right)\hat{k} \tag{6.110}$$

With $v_x = \dot{x} + \dot{\theta} z$ and $v_z = \dot{z} - x\dot{\theta}$ we can write this as

$$\vec{v} = v_x \hat{i} + \left(v_z + b\dot{\theta}\right) \hat{k} \tag{6.111}$$

The position of the slosh mass is

$$\vec{r}_f = (x - a\cos\psi)\hat{i} + (z + a\sin\psi)\hat{k} \tag{6.112}$$

so

$$\vec{v}_f = \dot{\vec{r}}_f - \omega^{\times}\vec{r}_f \tag{6.113}$$

or

$$\vec{v}_f = (\dot{x} + a\dot{\psi}\sin\psi)\hat{i} + (\dot{z} + a\dot{\psi}\cos\psi)\hat{k} + \dot{\theta}(a\cos\psi - x)\hat{k} + \dot{\theta}(z + a\sin\psi)\hat{i} \tag{6.114}$$

Rearranging

$$\vec{v}_f = (\dot{x} + a\dot{\psi}\sin\psi + \dot{\theta}(z + a\sin\psi))\hat{i} + (\dot{z} + a\dot{\psi}\cos\psi + \dot{\theta}(a\cos\psi - x)\hat{k} \tag{6.115}$$

or

$$\vec{v}_f = (\dot{x} + a(\dot{\theta} + \dot{\psi})\sin\psi + z\dot{\theta})\hat{i} + (\dot{z} + a(\dot{\psi} + \dot{\theta})\cos\psi - x\dot{\theta})\hat{k} \tag{6.116}$$

or

$$\vec{v}_f = (v_x + a(\dot{\theta} + \dot{\psi})\sin\psi)\hat{i} + (v_z + a(\dot{\psi} + \dot{\theta})\cos\psi)\hat{k} \tag{6.117}$$

The kinetic energy is

$$T = \frac{1}{2} m_0 \vec{r}^2 + \frac{1}{2} m_f \vec{r}_f^2 + \frac{1}{2} I_0 \dot{\theta}^2 + \frac{1}{2} I_f \dot{\theta}_f^2 \tag{6.118}$$

or

$$\begin{aligned} T = {} & \frac{1}{2} m_0 \left(v_x^2 + (v_z + b\dot{\theta})^2\right) \\ & + \frac{1}{2} m_f \left((v_x + a(\dot{\theta} + \dot{\psi})\sin\psi)^2 + (v_z + a(\dot{\psi} + \dot{\theta})\cos\psi)^2\right) \\ & + \frac{1}{2} I_0 \dot{\theta}^2 + \frac{1}{2} I_f (\dot{\theta} + \dot{\psi})^2 \end{aligned} \tag{6.119}$$

Expanding the squares

$$\begin{aligned} T = {} & \frac{1}{2} m_0 \left(v_x^2 + v_z^2 + 2v_z b\dot{\theta} + b^2\dot{\theta}^2\right) \\ & + \frac{1}{2} m_f \left(v_z^2 + 2v_z a(\dot{\psi} + \dot{\theta})\cos\psi\right) \end{aligned} \tag{6.120}$$

$$+\frac{1}{2}m_f\left(v_x^2+2v_x a(\dot{\theta}+\dot{\psi})\sin\psi\right)$$
$$+\frac{1}{2}m_f a^2(\dot{\theta}^2+2\dot{\psi}\dot{\theta}+\dot{\psi}^2)$$
$$+\frac{1}{2}I_0\dot{\theta}^2+\frac{1}{2}I_f(\dot{\theta}^2+2\dot{\theta}\dot{\psi}+\dot{\psi}^2)$$

The potential energy is due to the spring at the slosh hinge and is

$$V=\frac{1}{2}k\psi^2 \tag{6.121}$$

and the Raleigh dissipation function is

$$R=\frac{1}{2}c\dot{\psi}^2 \tag{6.122}$$

The equations are

$$0=(m_0+m_f)\dot{v}_x+m_f a(\ddot{\theta}+\ddot{\psi})\sin\psi+m_f a(\dot{\theta}+\dot{\psi})\dot{\psi}\cos\psi \tag{6.123}$$
$$0=(m_0+m_f)\dot{v}_z+m_f a(\ddot{\theta}+\ddot{\psi})\cos\psi-m_f a(\dot{\theta}+\dot{\psi})\dot{\psi}\sin\psi+m_0 b\ddot{\theta} \tag{6.124}$$
$$0=(I_0+I_f+m_f a^2)\ddot{\theta}+m_f a(\dot{v}_x\sin\psi+\dot{v}_z\cos\psi) \tag{6.125}$$
$$+m_f a\dot{\psi}(v_x\cos\psi-v_z\sin\psi)+(I_f+m_f a^2)\ddot{\psi}$$
$$0=(I_f+m_f a^2)(\ddot{\psi}+\ddot{\theta})+m_f a(v_x\cos\psi-v_x\sin\psi)\dot{\psi} \tag{6.126}$$
$$-a(\dot{\theta}+\dot{\psi})(v_z\sin\psi-v_x\cos\psi)$$

To the last equation, we add the damping and spring stiffness

$$M_\psi=-c\dot{\psi}-k\psi \tag{6.127}$$

Linearized equations

$$0=(m_0+m_f)\dot{v}_x \tag{6.128}$$
$$0=(m_0+m_f)\dot{v}_z+m_f a(\ddot{\theta}+\ddot{\psi})+m_0 b\ddot{\theta} \tag{6.129}$$
$$0=(I_0+I_f+m_f a^2)\ddot{\theta}+m_f a\dot{v}_z+(I_f+m_f a^2)\ddot{\psi} \tag{6.130}$$
$$0=(I_f+m_f a^2)(\ddot{\psi}+\ddot{\theta})+c\dot{\psi}+k\psi \tag{6.131}$$

An alternative is to use quasicoordinates [3,4]. Lagrange's equation for the spacecraft angle and velocities are found from

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\vec{v}}\right)+\omega^{\times}\frac{\partial L}{\partial\vec{v}}=0 \tag{6.132}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \vec{\omega}}\right) + \vec{\omega}^{\times}\frac{\partial L}{\partial \vec{\omega}} + \vec{v}^{\times}\frac{\partial L}{\partial \vec{v}} = 0 \tag{6.133}$$

where

$$\vec{a}^{\times} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{6.134}$$

Neither equation has the partial with respect to the generalized coordinate. The dynamical equations are

$$\begin{aligned} F_x(m_0 + m_f)\dot{v}_x + m_f a(\ddot{\theta} + \ddot{\psi})\sin\psi & \qquad (6.135)\\ + m_f a(\dot{\theta} + \dot{\psi})\dot{\psi}\cos\psi + m_f a(\dot{\theta} + \dot{\psi})^2\cos\psi & \\ F_z = (m_0 + m_f(\dot{v}_z - \dot{\theta}v_x) + m_f a(\ddot{\theta} + \ddot{\psi})\cos\psi & \qquad (6.136)\\ - m_f a(\dot{\theta} + \dot{\psi})\dot{\psi}\sin\psi & \\ + m_0 b\ddot{\theta} - m_f a(\dot{\theta} + \dot{\psi})^2\cos\psi & \\ M + bF_z = (I_0 + m_0 b^2)\ddot{\theta} + mb(\dot{v}_z - \dot{\theta}v_x) - c\dot{\psi} & \qquad (6.137)\\ 0 = (I_f + m_f a^2)(\ddot{\psi} + \ddot{\theta}) + m_f a(v_x\cos\psi - v_x\sin\psi)\dot{\psi} & \qquad (6.138)\\ - (\dot{\theta} + \dot{\psi})(v_z\sin\psi - v_x\cos\psi) + c\dot{\psi} + k\psi & \end{aligned}$$

6.10. N-body hub with single degree-of-freedom hinges

The N-body hub model can be specialized to a situation in which each hinge has only one degree-of-freedom. An example with two appendages is shown in Fig. 6.4.

The axes of rotation for the two appendages (the momentum wheel and telescope) are u_1 and u_2. The corresponding angular rate vectors are

$$\omega_0 = \omega_0 \tag{6.139}$$

$$\omega_k = B_k^T\omega_0 + \dot{\theta}_k u_k \tag{6.140}$$

where θ_k is the angle about the unit vector u_k, u_k is fixed in the body 0 frame, B_k transforms from the k frame to the 0 frame, and B_0 is the identity matrix. The corresponding accelerations are

$$\dot{\omega}_0 = \dot{\omega}_0 \tag{6.141}$$

$$\dot{\omega}_k = \dot{B}_k^T\omega_0 + B_k^T\dot{\omega}_0 + \ddot{\theta}_k u_k \tag{6.142}$$

The equations of the core attitude are

$$\dot{q} = f(\omega_0, q) \tag{6.143}$$

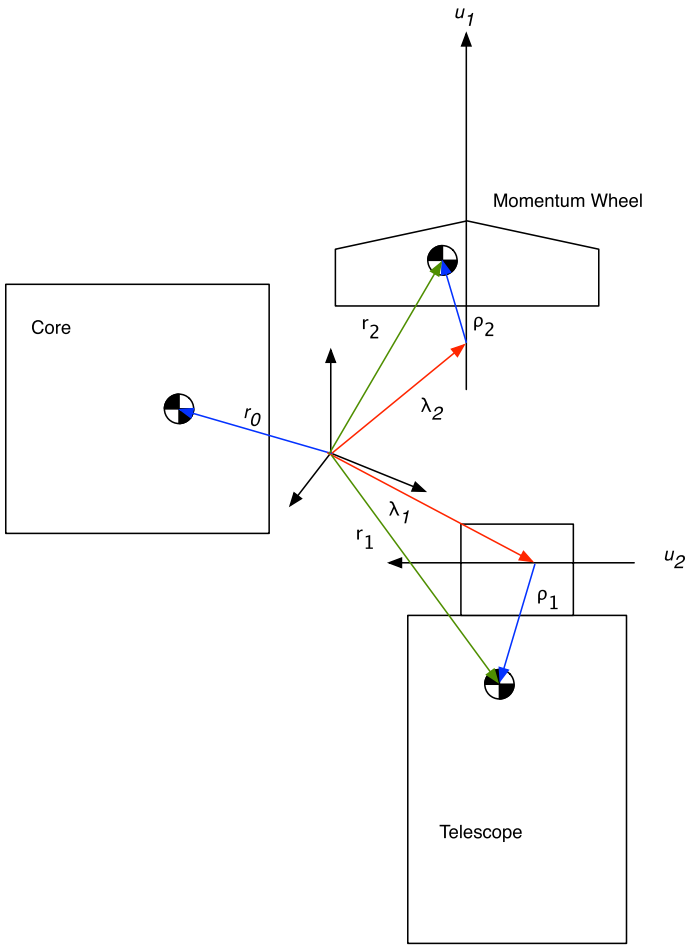


Figure 6.4 Dynamical model with two one-degree-of-freedom hinges for the momentum wheel and the telescope. Unlike a gyrostat, these are not necessarily rotating about their center-of-mass.

where q is the attitude quaternion from the ECI frame to the core body frame. Let B_k transform from the body k frame to the body 0 frame. Define

$$r_k = \lambda_k + B_k \rho_k \tag{6.144}$$

where A is the matrix that transforms from the 0 frame to the inertial frame and A^T corresponds to q.

$$H = A \sum_k B_k I_k \omega_k + \sum_k m_k D_k^{\times} \dot{D}_k \tag{6.145}$$

where I_k is the inertia matrix about the k body center-of-mass written in the k frame. This is conserved in the absence of external torques. Expanding the second term

$$H = A\sum_k \left(B_k I_k \omega_k + m_k d_k^\times (\omega_0^\times d_k + \dot{d}_k) \right) \tag{6.146}$$

Let

$$H = Ah \tag{6.147}$$

Then,

$$T = \omega_0^\times h + \dot{h} \tag{6.148}$$

Expanding

$$T = \omega_0^\times h + \sum_k \left(\dot{B}_k I_k \omega_k + B_k I_k \dot{\omega}_k + m_k (\dot{d}_k^\times \omega_0^\times d_k + d_k^\times \omega_0^\times \dot{d}_k - d_k^\times d_k^\times \dot{\omega}_0 + d_k^\times \ddot{d}_k) \right) \tag{6.149}$$

The angular-acceleration term is

$$\begin{array}{ll} \dot{\omega}_k = \dot{B}_k^T \omega_0 + B_k^T \dot{\omega}_0 + \ddot{\theta}_k u_k & k > 0 \\ \dot{\omega}_k = \dot{\omega}_0 & k = 0 \end{array} \tag{6.150}$$

where T is the torque on the entire spacecraft expressed in the 0 frame. The appendage equations are found by computing the reaction to the torques at the hinge points. The total angular momentum at the hinge is

$$H_k = AB_k I_k \omega_k + m_k (AB_k \rho_k)^\times \dot{D}_k \tag{6.151}$$

Expanding

$$H_k = Ah_k \tag{6.152}$$

$$h_k = B_k I_k \omega_k + m_k (B_k \rho_k)^\times (\omega_0^\times d_k + \dot{d}_k) \tag{6.153}$$

The scalar torque equation about u_k is

$$\begin{aligned} T_k = u_k^T [& \omega_0^\times h_k \\ & + \dot{B}_k I_k \omega_k + B_k I_k \dot{\omega}_k + m_k (\dot{B}_k \rho_k)^\times (\omega_0^\times d_k + \dot{d}_k) \\ & + m_k (B_k \rho_k)^\times (\omega_0^\times \dot{d}_k - d_k^\times \dot{\omega}_0 + \ddot{d}_k)] \end{aligned} \tag{6.154}$$

Expanding d_k and its derivatives

$$d_k = \lambda_k + B_k \rho_k - c \tag{6.155}$$

$$d_k = \sum_j s_{kj}(\lambda_j + B_j \rho_j) \tag{6.156}$$

$$\dot{d}_k = \sum_j s_{kj} B_j \Omega_j^{\times} \rho_j \tag{6.157}$$

$$\ddot{d}_k = \sum_j s_{kj} B_j (\Omega_j^{\times} \Omega_j^{\times} \rho_j - \rho_j^{\times} \dot{\Omega}_j) \tag{6.158}$$

$$s = \begin{bmatrix} 1-\mu_0 & -\mu_1 & -\mu_2 \\ -\mu_0 & 1-\mu_1 & -\mu_2 \\ -\mu_0 & -\mu_1 & 1-\mu_2 \end{bmatrix} \tag{6.159}$$

The final equation will be

$$\begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & a_{11} & a_{12} \\ A_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} \dot{\omega}_0 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} = \begin{bmatrix} E_0 \\ e_1 \\ e_2 \end{bmatrix} \tag{6.160}$$

where A_{00} is a 3×3 matrix and is the inertia matrix for the entire vehicle, A_{01} and A_{02} are 3×1 matrices, E_0 is 3×1, and e_1 and e_2 are scalars. The inertia matrix–vector and matrix terms are

$$A_{00} = I_0 + \sum_{k>0} \left(B_k I_k B_k^T - m_k d_k^{\times} d_k^{\times} \right) \tag{6.161}$$

The remaining terms in the inertia matrix are

$$\sum_k \left(B_k I_k u_k \ddot{\theta}_k - m_k d_k^{\times} \sum_j s_{kj} B_j \rho_j^{\times} u_j \ddot{\theta}_j \right) \tag{6.162}$$

$$A_{0k} = B_k I_k u_k - \sum_j s_{jk} m_j d_j^{\times} B_k \rho_k^{\times} u_k \tag{6.163}$$

The right-hand-side term is

$$E_0 = T - \omega_0^{\times} h - \sum_k \left[\left(\dot{B}_k I_k \omega_k + B_k I_k \dot{B}_k^T \omega_0 \right. \right. \tag{6.164}$$

$$\left. \left. + m_k \left(\dot{d}_k^{\times} \omega_0^{\times} d_k + d_k^{\times} \omega_0^{\times} \dot{d}_k + d_k^{\times} \sum_j s_{kj} B_j (\Omega_j^{\times} \Omega_j^{\times} \rho_j) \right) \right] \right. \tag{6.165}$$

where T is the external torque on the entire spacecraft.

$A_{01}^T = A_{10}$ and $a_{12} = a_{21}$ since the inertia matrix is symmetric.

$$A_{k0} = u_k^T [B_k I_k B_k^T - m_k (B_k \rho_k)^{\times} d_k^{\times}] \tag{6.166}$$

The scalar inertia terms for the appendages come from the expression

$$u_k^T(B_k I_k u_k \ddot{\theta}_k - m_k(B_k\rho_k)^\times \sum_j s_{kj} B_j \rho_j^\times u_j \ddot{\theta}_j) \tag{6.167}$$

$$a_{11} = u_1^T(B_1 I_1 u_1 - m_1(B_1\rho_1)^\times s_{11} B_1 \rho_1^\times u_1) \tag{6.168}$$
$$a_{12} = -m_1 s_{12} u_1^T (B_1\rho_1)^\times B_2 \rho_2^\times u_2 \tag{6.169}$$
$$a_{22} = u_2^T(B_2 I_2 u_2 - m_2(B_2\rho_2)^\times s_{22} B_2 \rho_2^\times u_2) \tag{6.170}$$
$$a_{21} = -m_2 s_{21} u_2^T (B_2\rho_2)^\times B_1 \rho_1^\times u_1 \tag{6.171}$$

Note that $m_1 s_{12} = m_2 s_{21}$. To show that $a_{12} = a_{21}$

$$u_1^T(B_1\rho_1)^\times B_2\rho_2^\times u_2 = ((B_1\rho_1)^\times B_2\rho_2^\times u_2)^T u_1 \tag{6.172}$$
$$= u_2^T((B_1\rho_1)^\times B_2\rho_2^\times)^T u_1 \tag{6.173}$$

Center-of-mass movement causes inertial coupling between the appendages. This disappears if $\rho_1 = 0$ or $\rho_2 = 0$. The right-hand-side terms contain all the remaining terms:

$$\begin{aligned} e_k = T_k - u_k^T[&\omega_0^\times \left(B_k I_k \omega_k + m_k(B_k\rho_k)^\times(\omega_0^\times d_k + \dot{d}_k)\right) \\ &+ \dot{B}_k I_k \omega_k + B_k I_k \dot{B}_k^T \omega_0 + m_k(\dot{B}_k\rho_k)^\times(\omega_0^\times d_k + \dot{d}_k) \\ &+ m_k(B_k\rho_k)^\times(\omega_0^\times \dot{d}_k + \omega_0^\times \omega_0^\times d_k + 2\omega_0^\times \dot{d}_k)] \end{aligned} \tag{6.174}$$

Checking symmetry in the inertia matrix is a good way to check the derivation. Angular-momentum conservation is shown in Example 6.6.

This model should agree with the gyrostat model. Example 6.7 shows that the two models do agree. The hub model propagates the wheel angles. This permits the incorporation of angle-dependent friction or other disturbances.

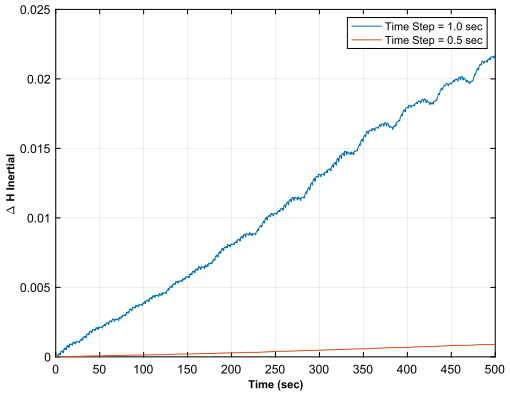
6.11. N-body hub with wheels

Fig. 6.5 shows a model of the James Webb Space Telescope. There are six reaction wheels, assumed to be ideal rotors and two appendages that are assumed to have three-degrees-of-freedom joints with respect to the core spacecraft. In this case, the appendages represent masses of fuel that move as a rigid body. The three degrees-of-freedom allow pendular motion and motion about the pendulum axis.

As a simplifying assumption, assume that the movement of fuel does not move the center-of-mass, that is $\dot{c} = 0$. The vector to the kth appendage body in the core frame and its derivatives are

$$d_k = \lambda_k + B_k\rho_k - c \tag{6.175}$$

```
d.inr        = [diag([1 2 3])  diag([1 2 1]) diag
       ([0.2 2 0.5])];
d.m          = [1 0.1 0.1];
d.u          = [[0;0;0] [0;1;0] [1;0;0]];
d.rho        = [[0;0;0] [0.1;0;0.3] [0;0;0.2]];
d.lambda     = [[0;0;0] [0.1;0.1;0] [0;.2;0.2]];
d.torque0    = [0;0;0]; d.torque = [0 0];
dT           = [0.5  0.25]; n = 1000;
x            = [1;0;0;0;.1;0.2;0.3;0;0;0.1;0.2];
h1           = zeros(1,n);
for k = 1:n
        [~,h]    = RHSNBodyCentralHub(x,0,d);
        h1(k)    = Mag(h);
        x        = RK4('RHSNBodyCentralHub',x,dT
                (1),0,d);
end
x = [1;0;0;0;.1;0.2;0.3;0;0;0.1;0.2];
h2 = zeros(1,2*n);
for k = 1:2*n
        [~,h]    = RHSNBodyCentralHub(x,0,d);
        h2(k)    = Mag(h);
        x        = RK4('RHSNBodyCentralHub',x,dT
                (2),0,d);
end
t1 = (0:(n  -1))*dT(1); t2 = (0:(2*n-1))*dT(2);
NewFig('|H|')
plot(t1,abs(h1-h1(1)),t2,abs(h2-h2(1)));
legend('Time Step = 1.0 sec','Time Step = 0.5 sec
       ');
XLabelS('Time (sec)'); YLabelS('\Delta H Inertial
       ');
grid
```



Example 6.6: Angular-momentum conservation for a three-body model with a single degree-of-freedom hinges at the attachment points. The growth of momentum changes with the size of the time step.

$$\dot{d}_k = B_k \Omega_k^{\times} \rho_k \tag{6.176}$$

$$\ddot{d}_k = B_k \Omega_k^{\times} \Omega_k^{\times} \rho_k - B_k \rho_k^{\times} \dot{\Omega}_k \tag{6.177}$$

where B_k is the transformation matrix from the kth frame to the core frame. The total angular momentum in the core-body frame is

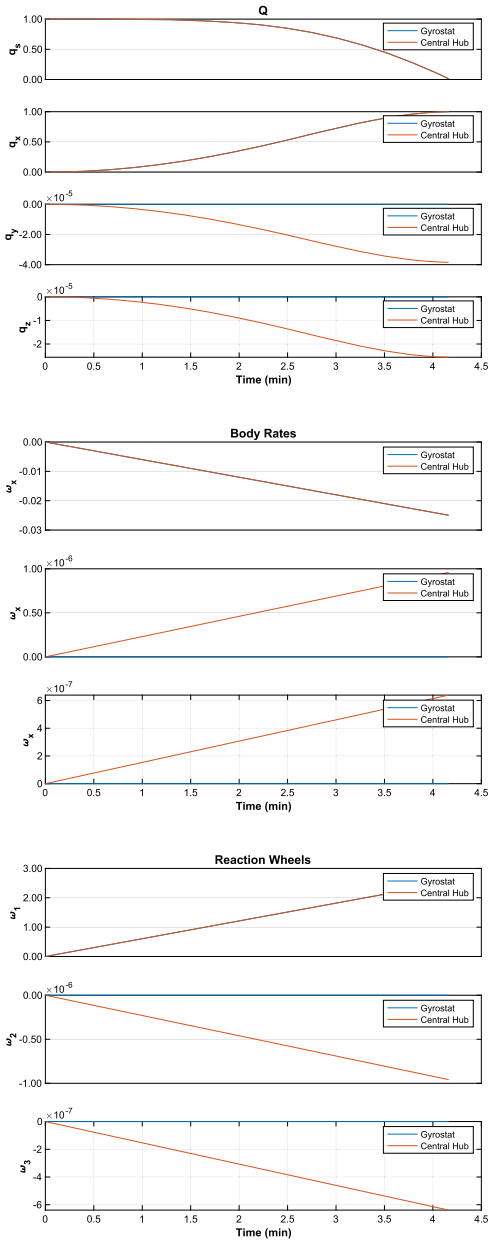
$$h = I_0 \omega_0 - m_0 d_0^{\times} d_0^{\times} + \sum_{k=1}^{N_w} u_k J_k (\Omega_k^w + u_k^T \omega_0) \tag{6.178}$$

$$+ \sum_{k=1}^{N_a} \left[B_k I_k (\Omega_k^s + B_k^T \omega_0) + m_k d_k^{\times} (\omega_0^{\times} d_k + \dot{d}_k) \right]$$

where J_k is the scalar reaction wheel inertia, u_k is the axis of rotation for the kth wheel, I_k is the appendage inertia, Ω_k is the appendage angular rate measured in the appendage frame, N_w is the number of wheels, and N_a is the number of appendages. Reorganizing

$$h = \left(I_0 - m_0 d_0^{\times} d_0^{\times} + \sum_{k=1}^{N_w} u_k J_k u_k^T + \sum_{k=1}^{N_a} \left[B_k I_k B_k^T - m_k d_k^{\times} d_k^{\times} \right] \right) \omega_0 \tag{6.179}$$

```
dCH.inr         = {diag([1 2 3]) ...
diag([0.01 0 0]) diag([0 0.01 0]) diag([0 0
      0.01])};
dCH.m           = [1 0.1 0.1 0.1];
dCH.u           = [0 1 0 0;0 0 1 0;0 0 0 1];
dCH.rho         = zeros(3,4);
dCH.lambda      = [0 0.1 0 0;0 0 0.1 0;0 0 0 0.1];
dCH.torque0     = [0;0;0];
dCH.torque      = [0 0 0];

dGS             = RHSGyrostat;
dGS.inr         = dCH.inr{1};
dT          = 0.25;
n           = 1000;
xCH         = [1;0;0;0; 0;0;0; 0;0;0; 0;0;0];
xGS         = [1;0;0;0; 0;0;0; 0;0;0];
xPCH        = zeros(13,n);
xPGS        = zeros(10,n);
dCH.torque(1) = 0.0001;
dGS.torqueWheel(1) = 0.0001;
for k = 1:n
  xPCH(:,k) = xCH;
  xPGS(:,k) = xGS;
  xCH = RK4('RHSNBodyCentralHub',xCH,dT,0,dCH);
  xGS = RK4('RHSGyrostat',xGS,dT,0,dGS);
end


yL = {'q_s' 'q_x' 'q_y' 'q_z' ...
      '\omega_x' '\omega_x' '\omega_x' ...
      '\omega_1' '\omega_2' '\omega_3'};

t = (0:n-1)*dT;
l = {'Gyrostat' 'Central Hub'};
k = 1:4;
pF = {[1 5] [2 6] [3 7] [4 8]};
TimeHistory(t,[xPGS(k,:);xPCH(k,:)],yL(k),'Q',pF
      ,{1 1 1 1});

k = 5:7;
pF = {[1 4] [2 5] [3 6]};
TimeHistory(t,[xPGS(k,:);xPCH(k,:)],yL(k),'Body
      Rates',pF,{1 1 1});

k = 8:10;
kC = k+3;
pF = {[1 4] [2 5] [3 6]};
TimeHistory(t,[xPGS(k,:);xPCH(kC,:)],yL(k),'
      Reaction Wheels',pF,{1 1 1});
```



Example 6.7: Comparison of the gyrostat and hub models.

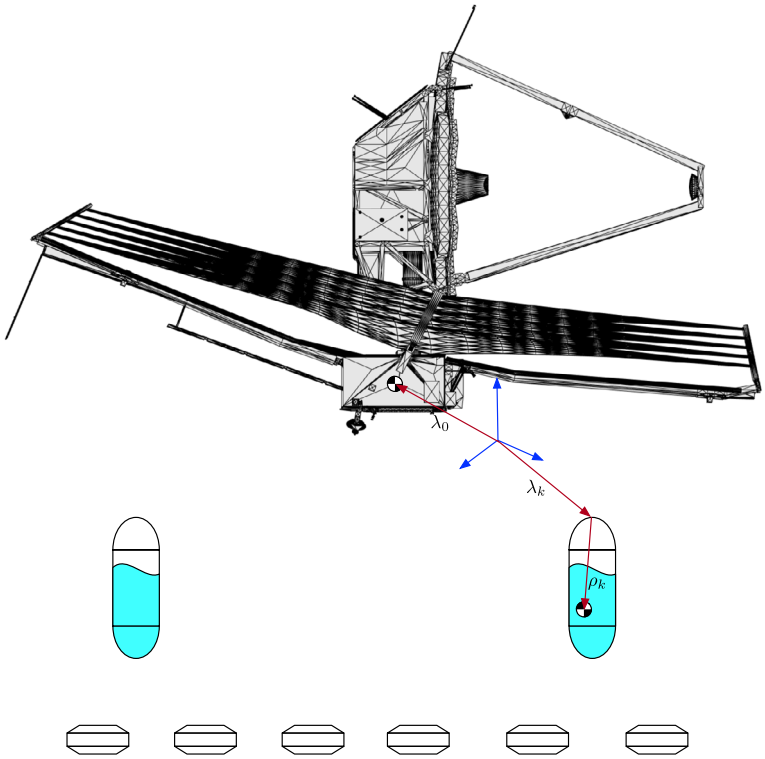


Figure 6.5 James Webb Space Telescope dynamical model. There is the core body, six reaction wheels assumed to be perfectly symmetric, and two fuel tanks.

$$+\sum_{k=1}^{N_w} u_k J_k \Omega_k^w + \sum_{k=1}^{N_a} \left[B_k I_k \Omega_k + m_k d_k^\times \dot{d}_k \right]$$

The dynamical equation for the core is

$$T = \omega_0^\times h + \dot{h} \tag{6.180}$$

or

$$\begin{aligned} T = I\dot{\omega}_0 &+ \left(\sum_{k=1}^{N_a} \left[\dot{B}_k I_k B_k^T + B_k I_k \dot{B}_k^T - m_k (\dot{d}_k^\times d_k^\times + d_k^\times \dot{d}_k^\times) \right] \right) \omega_0 \\ &+ \sum_{k=1}^{N_w} u_k J_k \dot{\Omega}_k^w + \sum_{k=1}^{N_a} \left[\dot{B}_k I_k \Omega_k + B_k I_k \dot{\Omega}_k + m_k d_k^\times (B_k \Omega_k^\times \Omega_k^\times \rho_k - B_k \rho_k^\times \dot{\Omega}_k) \right] \\ &+ \omega_0^\times h \end{aligned} \tag{6.181}$$

where

$$I = I_0 - m_0 d_0^{\times} d_0^{\times} + \sum_{k=1}^{N_w} u_k J_k u_k^T + \sum_{k=1}^{N_a} \left[B_k I_k B_k^T - m_k d_k^{\times} d_k^{\times} \right] \tag{6.182}$$

Rearranging to separate the inertia terms

$$\begin{aligned} T = {} & I\dot{\omega}_0 + \sum_{k=1}^{N_w} u_k J_k \dot{\Omega}_k^w + \sum_{k=1}^{N_a} \left(B_k I_k - m_k d_k^{\times} B_k \rho_k^{\times} \right) \dot{\Omega}_k \\ & + \sum_{k=1}^{N_a} \Big[\left(\dot{B}_k I_k B_k^T + B_k I_k \dot{B}_k^T - m_k (\dot{d}_k^{\times} d_k^{\times} + d_k^{\times} \dot{d}_k^{\times}) \right) \omega_0 \\ & + \dot{B}_k I_k \Omega_k + m_k d_k^{\times} B_k \Omega_k^{\times} \Omega_k^{\times} \rho_k \Big] \\ & + \omega_0^{\times} h \end{aligned} \tag{6.183}$$

These are the first three dynamical equations. The remaining equations are the reaction-wheel equations and the appendage equations. The reaction-wheel dynamical equations are

$$T_k = J_k (\dot{\Omega}_k + u_k \dot{\omega}_0) \tag{6.184}$$

The appendage dynamical equations are found by taking derivatives of the inertial angular momentum about the hinge.

$$H_k = A B_k I_k (\Omega_k + B_k^T \omega_0) + m_k (A B_k \rho_k)^{\times} \dot{D}_k \tag{6.185}$$

where

$$D_k = A d_k \tag{6.186}$$

$$\dot{D}_k = A(\omega_0^{\times} d_k + \dot{d}_k) \tag{6.187}$$

$$\ddot{D}_k = A \left(\omega_0^{\times} \omega_0^{\times} d_k + 2\omega_0^{\times} \dot{d}_k + \ddot{d}_k - d_k^{\times} \dot{\omega}_0 \right) \tag{6.188}$$

and

$$(A B_k \rho_k)^{\times} = A B_k \rho_k^{\times} B_k^T A^T \tag{6.189}$$

Collecting all the ω_0 terms

$$H_k = A B_k I_k (\Omega_k + B_k^T \omega_0) + m_k (A B_k \rho_k)^{\times} A (\dot{d}_k - d_k^{\times} \omega_0) \tag{6.190}$$

$$H_k = A B_k \left((I_k B_k^T - m_k \rho_k^{\times} B_k^T d_k^{\times}) \omega_0 + I_k \Omega_k + m_k \rho_k^{\times} B_k^T \dot{d}_k \right) \tag{6.191}$$

$$T_k = A^T B_k^T \dot{H}_k \tag{6.192}$$

Let

$$h_k = (I_k B_k^T - m_k \rho_k^\times B_k^T d_k^\times)\omega_0 + I_k \Omega_k + m_k \rho_k^\times B_k^T \dot{d}_k \tag{6.193}$$

The derivative of H_k is

$$\begin{aligned} T_k = & \left(B_k^T \omega_0^\times B_k + \Omega_k^\times\right) h_k \\ & + (I_k \dot{B}_k^T - m_k \rho_k^\times \dot{B}_k^T d_k^\times - m_k \rho_k^\times B_k^T \dot{d}_k^\times)\omega_0 + + m_k \rho_k^\times \dot{B}_k^T \dot{d}_k \\ & + (I_k B_k^T - m_k \rho_k^\times B_k^T d_k^\times)\dot{\omega}_0 + I_k \dot{\Omega}_k \\ & + m_k \rho_k^\times B_k^T \left(B_k \Omega_k^\times \Omega_k^\times \rho_k - B_k \rho_k^\times \dot{\Omega}_k\right) \end{aligned} \tag{6.194}$$

Rearranging to separate the inertia terms

$$\begin{aligned} T_k = & \left(I_k - \rho_k^\times \rho_k^\times\right) \dot{\Omega}_k + (I_k B_k^T - m_k \rho_k^\times B_k^T d_k)\dot{\omega}_0 + \left(\omega_0^\times B_k + \Omega_k^\times\right) h_k \\ & + (I_k \dot{B}_k^T - m_k \rho_k^\times \dot{B}_k^T d_k^\times - m_k \rho_k^\times B_k^T \dot{d}_k^\times)\omega_0 + m_k \rho_k^\times \dot{B}_k^T \dot{d}_k \\ & + m_k \rho_k^\times \Omega_k^\times \Omega_k^\times \rho_k \end{aligned} \tag{6.195}$$

The generalized inertia matrix, for one wheel and one appendage, where body 2 is the wheel and body 3 is the appendage, is

$$G = \begin{bmatrix} I & u_k J_k & B_k I_k - m_k d_k^\times B_k \rho_k^\times \\ u_k^T J_k & J_k & 0 \\ I_k B_k^T - m_k \rho_k^\times B_k^T d_k^\times & 0 & I_k - m_k \rho_k^\times \rho_k^\times \end{bmatrix} \tag{6.196}$$

There is no inertial coupling between the appendages or the wheels. The (1,3) term should be the transpose of the (3,1) term.

$$\begin{aligned} (B_k I_k)^T &= I_k^T B_k^T = I_k B_k^T \\ (d_k^\times B_k \rho_k^\times)^T &= (B_k \rho_k^\times)^T (d_k^\times)^T = (\rho_k^\times)^T B_k^T (d_k^\times)^T = \rho_k^\times B_k^T d_k^\times \end{aligned} \tag{6.197}$$

which agree.

6.12. Control-moment gyros

This section derives the equations for a spacecraft with N single-axis control-moment gyros (CMGs). The CMG is shown in Fig. 6.6. The momentum wheel is assumed to be perfectly symmetric. The axis of gimbaling is aligned with its center-of-mass. Thus, rotating the CMG does not move the spacecraft's center-of-mass. CMGs should not be confused with spinning wheels on gimbaled platforms that are used for sensing.

The CMGs are numbered $k = 1$ through N. Each CMG has its base frame and its gimbaled frame. The base-to-spacecraft transformation matrix is B_{0_k}. The gimbal-to-base transformation matrix is B_{g_k}. B_{0_k} is constant. Body 0 is the core spacecraft. The

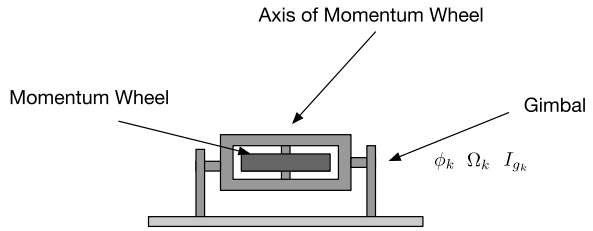


Figure 6.6 Single-gimbal control-moment gyro (CMG). The momentum wheel spins at a constant angular rate.

gimbal matrix for CMG k is

$$B_{g_k} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi_k & -\sin\phi_k \\ 0 & \sin\phi_k & \cos\phi_k \end{bmatrix} \tag{6.198}$$

The gimbal axis is always the x-axis. The gimbal time derivative matrix is

$$\dot{B}_{g_k} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin\phi_k & -\cos\phi_k \\ 0 & \cos\phi_k & -\sin\phi_k \end{bmatrix} \dot{\phi}_k \tag{6.199}$$

The external torque on the core spacecraft is T. The torque on gimbal k is T_{g_k}. The torque on wheel k is T_{w_k}. See Table 6.2 for all definitions of symbols used in the CMG derivation.

We write the total angular momentum in the inertial frame.

$$H = A\left(I_0\omega + \sum_{k=1}^{N} B_{0_k}\left[I_{g_k}(\Omega_{g_k} + B_{0_k}^T\omega) + B_{g_k}I_{w_k}(\Omega_{w_k} + B_{g_k}^T B_{0_k}^T\omega + B_{g_k}^T\Omega_{g_k})\right]\right) \tag{6.200}$$

Let

$$h_k = I_{g_k}(\Omega_{g_k} + B_{0_k}^T\omega) + B_{g_k}I_{w_k}(\Omega_{w_k} + B_{g_k}^T B_{0_k}^T\omega + B_{g_k}^T\Omega_{g_k}) \tag{6.201}$$

Taking the derivative with respect to time and multiplying by A^T we get

$$T = I_0\dot{\omega} + \omega^\times\left(I_0\omega + \sum_{k=1}^{N} B_{0_k}h_k\right) + \sum_{k=1}^{N} B_{0_k}\dot{h}_k \tag{6.202}$$

and

$$\begin{aligned}\dot{h}_k = {} & I_{g_k}(\dot{\Omega}_{g_k} + B_{0_k}^T\dot{\omega}) + \dot{B}_{g_k}I_{w_k}(\Omega_{w_k} + B_{g_k}^T B_{0_k}^T\omega + B_{g_k}^T\Omega_{g_k}) \\ & + B_{g_k}I_{w_k}(\dot{\Omega}_{w_k} + \dot{B}_{g_k}^T B_{0_k}^T\omega + B_{g_k}^T B_{0_k}^T\dot{\omega} + \dot{B}_{g_k}^T\Omega_{g_k} + B_{g_k}^T\dot{\Omega}_{g_k})\end{aligned} \tag{6.203}$$

Table 6.2 CMG model symbols.

Symbol	Name	Units
B_{g_k}	Transformation matrix from the gimbal k to the base k frame	none
B_{0_k}	Transformation matrix from the CMG k base frame to the core 0 frame	none
T	External torque on the spacecraft	Nm
T_{g_k}	Torque on gimbal k	Nm
T_{w_k}	Torque on wheel k	Nm
I_0	Core spacecraft inertia	kg-m^2
I_{g_k}	Gimbal k inertia	kg-m^2
I_{w_k}	Wheel k inertia	kg-m^2
ϕ_k	Gimbal k angle	rad
Ω_{w_k}	Wheel k angular rate	rad/s
Ω_{g_k}	Gimbal k angular rate	rad/s
u_{g_k}	Unit vector along the gimbal rotational axis	none
u_{w_k}	Unit vector along the wheel rotational axis	none
A	Transformation matrix from the spacecraft body frame to the inertial frame	none
ω	Body rate in the body frame	rad/s

Note that

$$\dot{B}_{g_k} = B_{g_k}\Omega_{g_k}^{\times} \tag{6.204}$$

$$\dot{A} = A\omega^{\times} \tag{6.205}$$

The gimbal-torque equations are

$$T_{g_k} = u_{g_k}\dot{h}_k \tag{6.206}$$

The wheel-torque equations are

$$T_{w_k} = u_{w_k}I_{w_k}(\dot{\Omega}_{w_k} + \dot{B}_{g_k}^T B_{0_k}^T\omega + B_{g_k}^T B_{0_k}^T\dot{\omega} + \dot{B}_{g_k}^T\Omega_{g_k} + B_{g_k}^T\dot{\Omega}_{g_k}) \tag{6.207}$$

To which we add the kinematic equations

$$\dot{q} = f(q,\omega) \tag{6.208}$$

$$\dot{\phi}_k = \Omega_{g_k} \tag{6.209}$$

where q is the quaternion from the inertial frame to the spacecraft frame and u_k is the unit vector along the axis of rotation.

The dynamical equations need to be assembled in the form

$$T = I(\phi)\dot{\omega} + g(\omega,\phi) \tag{6.210}$$

where I is the generalized inertia matrix and, in this case, ω is a vector of all angular rates that is of length $3+2N$. We rearrange $\dot{h}_k$ to make the division clearer

$$\begin{aligned}\dot{h}_k = &\left(I_{g_k}B_{0_k}^T + B_{g_k}I_{w_k}B_{g_k}^T B_{0_k}^T\right)\dot{\omega} \\ &+ (I_{g_k} + B_{g_k}I_{w_k}B_{g_k}^T)\dot{\Omega}_{g_k} + B_{g_k}I_{w_k}\dot{\Omega}_{w_k} \\ &+ \dot{B}_{g_k}I_{w_k}(\Omega_{w_k} + B_{g_k}^T B_{0_k}^T\omega + \Omega_{g_k}) \\ &+ B_{g_k}I_{w_k}\dot{B}_{g_k}^T\left(B_{0_k}^T\omega + \Omega_{g_k}\right)\end{aligned} \tag{6.211}$$

We assemble the inertia matrix from this equation. It should be symmetric. Looking at only one CMG we get

$$I = \begin{bmatrix} I_0 + B_{0_k}\left(I_{g_k} + B_{g_k}I_{w_k}B_{g_k}^T\right)B_{0_k}^T & B_{0_k}(I_{g_k} + B_{g_k}I_{w_k}B_{g_k}^T)u_{g_k} & B_{0_k}B_{g_k}I_{w_k}u_{w_k} \\ u_{g_k}^T\left(I_{g_k} + B_{g_k}I_{w_k}B_{g_k}^T\right)B_{0_k}^T & u_{g_k}^T\left(I_{g_k} + B_{g_k}I_{w_k}B_{g_k}^T\right)u_{g_k} & u_{g_k}^T B_{g_k}I_{w_k}u_{w_k} \\ u_{w_k}^T I_{w_k}B_{g_k}^T B_{0_k}^T & u_{w_k}^T I_{w_k}B_{g_k}^T u_{g_k} & u_{w_k}^T I_{w_k}u_{w_k} \end{bmatrix} \tag{6.212}$$

The inertia matrix is symmetric, as it should be. Note that there is no inertia coupling between the appendage and the reaction wheels. A simulation of a reaction-wheel torque pulse is shown in Example 6.8. The angular momentum is shown at two different timesteps. The number should be zero. It is lower when the time step is smaller, indicating that angular momentum is conserved. Both bodies react to the reaction-wheel torque. If we were modeling slosh, there would be a damper at the hinge of the second body.

6.13. Flexible structures

The simplest model of a flexible structure is shown in Fig. 6.7. The spacecraft is assumed to be composed of a rigid core to which a flexible structure is attached. The mass of the core is much larger than the mass of the structure, and the movement of the structure relative to the core is assumed to be small. Define

$$D = A(\lambda + r + \rho) \tag{6.213}$$

where the subscripts on r and ρ have been dropped. Taking derivatives

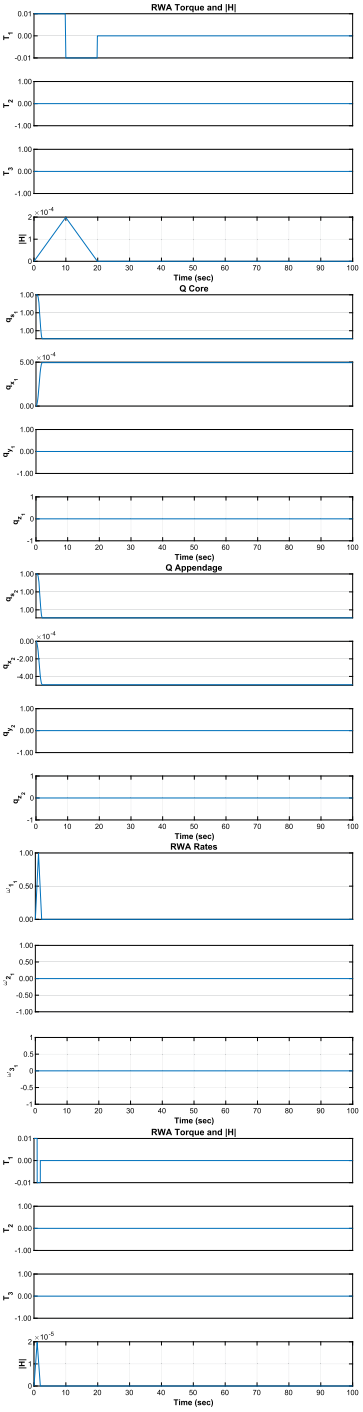
$$\dot{D} = A\omega^{\times}(\lambda + r + \rho) + A\dot{\rho} \tag{6.214}$$

$$\ddot{D} = A(\omega^{\times}\omega^{\times}(\lambda + r + \rho) + 2\omega^{\times}\dot{\rho} - (\lambda + r + \rho)^{\times}\dot{\omega} + \ddot{\rho} \tag{6.215}$$

The total angular momentum of the system is

$$H = AI\omega + mD^{\times}\dot{D} \tag{6.216}$$

```
%% Checks two time steps
% Can be configured to have just reaction wheels
rWAOnly = false;
nPulse  = 100;
dRHS    = RHSGyroNBody;
tEnd    = 100;
dT      = 0.01;
q       = [1;0;0;0];
if( rWAOnly)
  dRHS.nB = 1;
  x  = [q;0;0;0;0;0;0];
  m  = 14;
else
  dRHS.nB = [];
  x  = [q;q;0;0;0;0.0;0;0;0;0;0];
  m  = 21;
end
n       = floor(tEnd/dT);
torqueW = [0.01;0;0]; % Internal torque
xP      = zeros(m,n);
for k = 1:n
  if( k < nPulse )
    dRHS.torqueW = torqueW;
  elseif( k > nPulse && k < 2*nPulse )
    dRHS.torqueW = -torqueW;
  else
    dRHS.torqueW = [0;0;0];
  end
  [~,hX]  = RHSGyroNBody(x,0,dRHS);
        xP(:,k) = [x;dRHS.torqueW;Mag(hX)];
  x       = RK4('RHSGyroNBody',x,dT,0,dRHS);
end

t = (0:n-1)*dT;

yL = {'q_{s_1}' 'q_{x_1}' 'q_{y_1}' 'q_{z_1}'...
      'q_{s_2}' 'q_{x_2}' 'q_{y_2}' 'q_{z_2}'...
      '\omega_{x_1}' '\omega_{y_1}' '\omega_{x_1}'
            ,...
      '\omega_{x_2}' '\omega_{y_2}' '\omega_{z_2}'
            ,...
      '\omega_{1_1}' '\omega_{2_1}' '\omega_{3_1}'
            ,...
      'T_1' 'T_2' 'T_3' '|H|'};

k = 1:4;
TimeHistory(t,xP(k,:),yL(k),'Q Core');

if( m == 21 )
  k = 5:8;
  TimeHistory(t,xP(k,:),yL(k),'Q Appendage');
  k = 9:14;
  TimeHistory(t,xP(k,:),yL(k),'Rates');
  k = 15:17;
  TimeHistory(t,xP(k,:),yL(k),'RWA Rates');
  k = 18:21;
  TimeHistory(t,xP(k,:),yL(k),'RWA Torque and |H|
        ');
else
  k = 5:10; j = [9:11 15:17];
  TimeHistory(t,xP(k,:),yL(j),'Rates');
  k = 11:14; j = 18:21;
  TimeHistory(t,xP(k,:),yL(j),'RWA Torque and |H|
        ');
end

Figui
```



Example 6.8: Reaction-wheel pulse response for a spacecraft with two bodies and three reaction wheels.

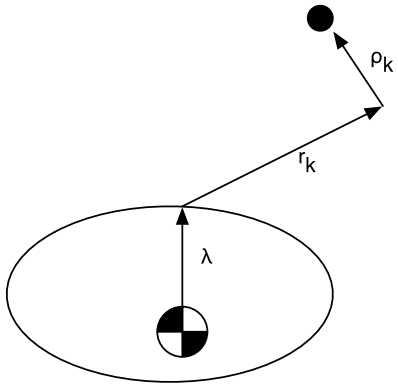


Figure 6.7 Flexible structure schematic.

where A transforms from the body to the inertial frame. The equations of motion of the entire spacecraft are

$$T = I\dot{\omega} + \omega^{\times} I\omega + m(\lambda + r + \rho)^{\times}(\omega^{\times}\omega^{\times}(\lambda + r + \rho) + 2\omega^{\times}\dot{\rho} - (\lambda + r + \rho)^{\times}\dot{\omega} + \ddot{\rho}) \quad (6.217)$$

If ρ is much smaller than r we can drop it from the $(\lambda + r + \rho)$ terms and keep only the derivative terms,

$$T = I\dot{\omega} + \omega^{\times} I\omega + m(\lambda + r)^{\times}(\omega^{\times}\omega^{\times}(\lambda + r) + 2\omega^{\times}\dot{\rho} - (\lambda + r)^{\times}\dot{\omega} + \ddot{\rho}) \quad (6.218)$$

Collecting terms,

$$T = I_T\dot{\omega} + \omega^{\times} I_T\omega + m(\lambda + r)^{\times}(2\omega^{\times}\dot{\rho} + \ddot{\rho}) \quad (6.219)$$

$$I_T = I - m(\lambda + r)^{\times}(\lambda + r)^{\times} \quad (6.220)$$

These equations are nonlinear in both body angular rate, ω, and mass displacement, ρ. The inertia matrix is constant because the flex deformation is assumed small.

The equation of motion for the mass is

$$F = m(\omega^{\times}\omega^{\times}(\lambda + r) + 2\omega^{\times}\dot{\rho} - (\lambda + r) \times \dot{\omega} + \ddot{\rho}) \quad (6.221)$$

where F are all of the forces acting on the mass. The coupled equations are

$$\begin{bmatrix} I_T & m(\lambda + r)^{\times} \\ -m(\lambda + r)^{\times} & m \end{bmatrix} \begin{bmatrix} \dot{\omega} \\ \ddot{\rho} \end{bmatrix} = \begin{bmatrix} \omega^{\times} I_T\omega \\ m(\omega^{\times}\omega^{\times}(\lambda + r) + 2\omega^{\times}\dot{\rho}) \end{bmatrix} + \begin{bmatrix} T \\ f \end{bmatrix} \quad (6.222)$$

Linearization of these equations must be done with care. If both ω and $\dot{\rho}$ are small then they become

$$\begin{bmatrix} I_T & m(\lambda+r)^\times \\ -m(\lambda+r)^\times & m \end{bmatrix} \begin{bmatrix} \dot{\omega} \\ \ddot{\rho} \end{bmatrix} = \begin{bmatrix} T \\ f \end{bmatrix} \tag{6.223}$$

The coupling is entirely through the inertia matrix. If I_T is large and the rates are small the equations become

$$\begin{bmatrix} I_T & m(\lambda+r)^\times \\ -m(\lambda+r)^\times & m \end{bmatrix} \begin{bmatrix} \dot{\omega} \\ \ddot{\rho} \end{bmatrix} = \begin{bmatrix} \omega^\times I_T \omega \\ m\omega^\times \omega^\times (\lambda+r) \end{bmatrix} + \begin{bmatrix} T \\ f \end{bmatrix} \tag{6.224}$$

The mass still has a ω^2 term driving it but ρ only appears on the left-hand side. This derivation can be extended to multiple bodies. First, rewrite the equations in the form

$$T = I\dot{\omega} + \omega^\times I\omega - \sum_k (\lambda + r_k)^\times f_k \tag{6.225}$$

$$f_k = m(\omega^\times \omega^\times (\lambda + r_k) - (\lambda + r_k)^\times \dot{\omega} + \ddot{\rho} \tag{6.226}$$

$$f_k = \sum_j k_{kj}(\rho_j - \rho_k) \tag{6.227}$$

where k_{kj} is the spring stiffness between j and k. Note that I is now just the core inertia. Just writing the equations for the masses

$$M\ddot{\rho} + K\rho = F \tag{6.228}$$

where K is symmetric and full. We define the following variables

$$\begin{aligned} \rho &= \Phi\eta \\ \Phi^T K \Phi &= \Omega^2 \\ \Phi^T M \Phi &= E \end{aligned}$$

where E is the identity matrix. Since ρ is a vector quantity, K is $3n$-by-$3n$. If the number of modes is m, F is $3n$ by m. The mass equations become

$$\ddot{\eta} + \Omega^2 \eta = \Phi^T F \tag{6.229}$$

where F contains the force terms dependent on ω, Φ is known as the modal transformation matrix in finite-element models, η are the modal amplitudes, and Ω are the mode frequencies.

References

[1] Z. Hussain, N. Azlan, KANE's method for dynamic modeling, in: 2016 IEEE International Conference on Automatic Control and Intelligent Systems, 10 2016, pp. 174–179.
[2] C.M. Roithmayr, Contribution of zonal harmonics to gravitational moment, Journal of Guidance, Control, and Dynamics 14 (1) (1991) 210–214.
[3] M. Reyhanoglu, Modeling and Control of Space Vehicles with Fuel Slosh Dynamics, Tech. Rep., Embry-Riddle Aeronautical University, February 2011.
[4] S. Cho, M. McClamroch, M. Reyhanoglu, Feedback control of a space vehicle with unactuated fuel slosh dynamics, in: AIAA Guidance, Navigation, and Control Conference and Exhibit, 2012.

CHAPTER 7

Environment

7.1. Space story

The Pioneer 10 and 11 spacecraft were the first deep-space missions. The spacecraft were launched in 1972 and 1973 and were the first to pass through the asteroid belt. In the 1980s it was observed that both spacecraft were slowing. Some scientists speculated that Einstein's General Theory of Relativity was incorrect. The problem was eventually traced by Slava Turyshev of JPL to thermal radiation from the spacecraft. The heat, like any radiation, produced a thrust decelerating the two probes [1].

7.2. Introduction

The space environment is all of the external influences on the spacecraft [2]. The environment is the source of force and torque disturbances on the spacecraft. It also is a source of radiation that damages electronics and plasma that can charge the spacecraft. The latter effects do not have a direct influence on spacecraft control but indirectly influence the control-system design as they drive requirements for more robust electronics.

This chapter covers the space environment. The disturbance effects of the space environment are covered in the disturbances chapter.

7.3. Optical environment

The optical environment includes all electromagnetic frequencies from infrared wavelengths to visible bands. Sources of optical influx are the Sun, planets, and other spacecraft. X-rays and gamma-rays are considered ionizing radiation sources.

7.3.1 Solar radiation

Solar pressure is the dominant disturbance on a spacecraft in geosynchronous and interplanetary orbit. It is also used for propulsion by solar sails. Solar pressure is due to the force of photons on the surfaces of the spacecraft. The pressure is

$$s = \frac{p}{c} \tag{7.1}$$

where p is the power of the sunlight and c is the speed of light. As can be seen, a 1-N force would require 3×10^8 W. Thus a flashlight-propelled spacecraft would take a long

https://doi.org/10.1016/B978-0-32-399915-1.00018-8

time to gain any velocity. The solar flux is

$$p = \frac{1367}{\left(\frac{r}{r_e}\right)^2} \tag{7.2}$$

where r_e is the mean distance of the Earth from the Sun. The units are W/m^2.

7.3.2 Earth albedo

Albedo is the reflection of sunlight off of the surface and clouds of the Earth. As the Earth cannot be treated as a point source (as can the Sun) and the reflectivity varies over the surface, albedo is difficult to model accurately [3]. The simplest model is to assume that reflection from the Earth's surface and clouds is diffuse [4]. The flux from the total reflectance can then be computed by integrating it over the surface of the Earth.

The Earth's surface is modeled as a Lambertian reflector as shown in Fig. 7.1.

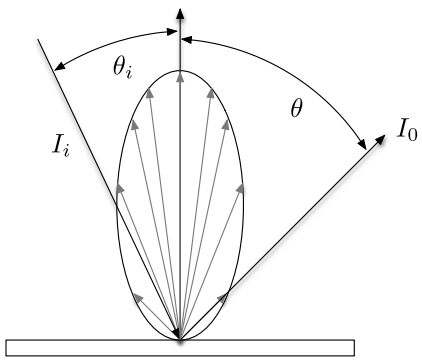


Figure 7.1 Lambertian reflector.

If the spacecraft is more than five times the distance from the surface as the dimensions of the surface, then the following equation can be used to model the flux [5]:

$$\Phi_{ij} = \alpha PA \frac{\cos\theta_i \cos\theta_j}{\pi r^2} \tag{7.3}$$

where α is the absorption coefficient for the thermal flux, P is the solar flux, A is the area of the surface patch, $\cos\theta_i$ is the angle between the normal of the surface patch and the incoming flux, and $\cos\theta_j$ is the angle between the normal of the spacecraft patch and the incoming radiation. For a spacecraft in a 200-km orbit, this would mean that the triangles representing the surface of the Earth could be no more than 40 km across.

This requires an active subdivision of the Earth's surface so that only the triangles under the spacecraft are tessellated. The α coefficient varies with each patch and may

represent ice, water, vegetation, clouds, etc. The spot under the spacecraft was subdivided into smaller triangles. The size of the triangles decreases with angular distance from the spacecraft vector.

7.3.3 Earth radiation

Earth radiation is assumed uniform over the surface of the Earth. The flux is

$$q = \frac{400}{\left(\frac{r}{r_E}\right)^2} \tag{7.4}$$

7.4. Atmosphere

The atmosphere interacts with the spacecraft producing disturbance forces and torques. In low-Earth orbit, drag compensation may be necessary to prevent premature reentry. It may also be necessary to maintain the desired phasing in a constellation.

Many models are used for atmospheric density. The simplest is an exponential atmosphere. This covers the entire atmosphere with a single equation but cannot fit atmospheric-density variations accurately. For the Earth, the exponential atmosphere model is

$$\rho = 1.225e^{-0.0817h^{1.15}} \tag{7.5}$$

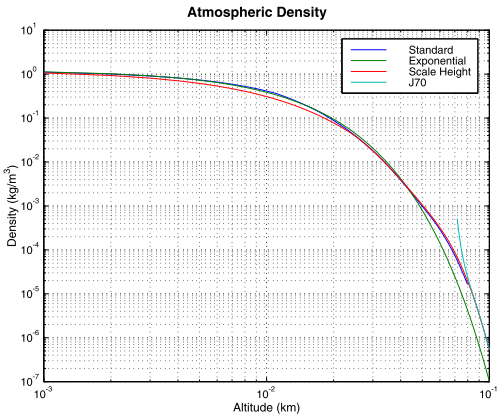
where h is the altitude in km. This is a variation on a true exponential model that is

$$\rho = \rho_0 e^{-\frac{h}{h_s}} \tag{7.6}$$

where h_s is the scale height and ρ_0 is the surface density. This can be expanded to a scale-height model. This is an exponential model that fits the model at many altitudes. However, the atmosphere does not follow an exponential law in all ranges of altitudes. The Sun has a major influence on the atmosphere and the density seen by the spacecraft is a function of the angle between the Sun and the spacecraft. A model that includes this effect is the Jacchia atmosphere model but it is only valid for altitudes > 70 km. The last is the standard atmosphere that is only valid under 18.6 km. The last curve fits actual data and is the best for atmosphere studies. The Jacchia model includes the effect of the Sun that is important for low-Earth orbiting satellites. The models are compared in Example 7.1.

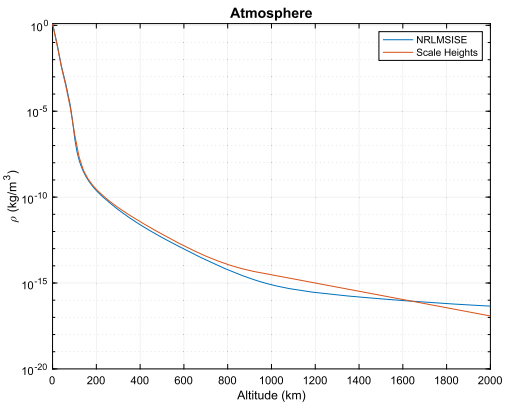
Other models exist including the MSIS model [6]. The NRLMSISE-00 model was developed by Mike Picone, Alan Hedin, and Doug Drob. It is based on extensive data from spacecraft. Example 7.2 compares it with scale heights. The results are close but scale heights do not include the Sun effects that can be important.

```
altitude = linspace(0,100);
z = StdAtm( altitude*1000 ); % standard
d = zeros(4,100);
k = 1:length(z.density);
d(1,k) = z.density;
% exponential
d(2,:) = AtmDens1(altitude);
% scale heights
d(3,:) = AtmDens2(altitude);
p = struct('aP', 400, 'dd', 79, ...
           'f', 230, 'fHat', 230, ...
           'fHat400',230,'lat',-30,...
           'lng', 0, 'mm', 840,...
           'yr', 1970 );
j = find( altitude >= 71 );
for k = j   % Jacchia 1970
  p.z = altitude(k);
  [rho, nHe, nN2, nO2, nO, tZ, eM] ...
        = AtmJ70( p );
  d(4,k) = rho*1000;
end
Plot2D( altitude/1000, d, ...
  'Altitude (km)','Density (kg/m^3)',...
  'Atmospheric Density', 'ylog')
legend('Standard', 'Exponential', ...
       'Scale Height', 'J70' )
```



Example 7.1: Atmosphere-density models.

```

%% NRL Model

for i = 1:8
        aph.a(i)=100;
end
input(1) = struct('year',0,'doy',172,'sec',29000,
       'alt',0,'lst',0,'g_lat',60,'g_long'
       ,-70,...
  'f107A',150,'f107',150,'ap',4,'ap_a',aph);


z     = zeros(1,24);
flags = struct('switches',z,'sw',z,'swc',z);

flags.switches(1) = 0;
flags.switches(10)=1; % Does not use daily ap

for i = 2:24
        flags.switches(i)=1;
end

alt = linspace(0,2000);
rho = zeros(1,length(alt));
input(1).ap_a = aph;
for k = 1:length(alt)
  input(1).alt = alt(k);
  flags.switches(1) = 0;
        output = AtmNRLMSISE( 'gtd7', [], input
              (1), flags );
  rho(k) = output.d(6);
end

rho2 = AtmDens2( alt );

Plot2D(alt,[rho*1000;rho2],'Altitude (km)','\rho
       (kg/m^3)','Atmosphere','ylog')

legend('NRLMSISE','Scale Heights')
```



Example 7.2: Comparison of NRLMSISE with scale heights.

The most accurate model is the Air Force "High Accuracy Satellite Drag Model" (HASDM) [7]. This model uses satellite orbit data to refine the model. It is a great improvement over the models cited above. As Bowman states [8]

> *Current thermospheric density models do not adequately account for dynamic changes in atmospheric drag for orbit predictions, and no significant operational improvements have been made since 1970. Lack of progress is largely due to poor model inputs in the form of crude heating indices, as well as poor model accuracy and resolution, both spatial and temporal.*

The European Space Agency has a similar initiative that might also be a source of improved models [9].

For planetary missions, the atmospheres of the target planets must be known. Densities for Titan and Neptune are shown in Fig. 7.2.

7.5. Plasma

A plasma is a gas in which the atoms are stripped of their electrons to that they have a net positive charge. The gas comprises both the ions and the electrons so that globally it is neutral. A spacecraft will collect charge and gain an electrostatic potential [10]. Charging can lead to surface arcing that can destroy electronics. For this reason, attitude control electronics should be located so arcs do not touch the electronics. This can be a problem for sensors that need to be located on the surface of the spacecraft. Design techniques to minimize differential charging are well understood and need to be employed [13], [11].

The attitude control system should have procedures in place in case a major component is lost due to charging. For example, a ground control loop can substitute for a damaged reaction wheel or flight computer.

7.6. Gravity

Gravity influences the orbit of the spacecraft and creates the gravity-gradient torque on the spacecraft. The following sections discuss gravity models. Gravity gradient is discussed in the chapter on disturbances.

7.6.1 Point mass

The point-mass model assumes that the mass is concentrated at an infinitesimal point at the center of the planet. The gravitational acceleration is

$$a = -\mu \frac{r}{|r|^3} \tag{7.7}$$

where μ is the gravitational parameter and r is the vector from the point mass to the spacecraft.

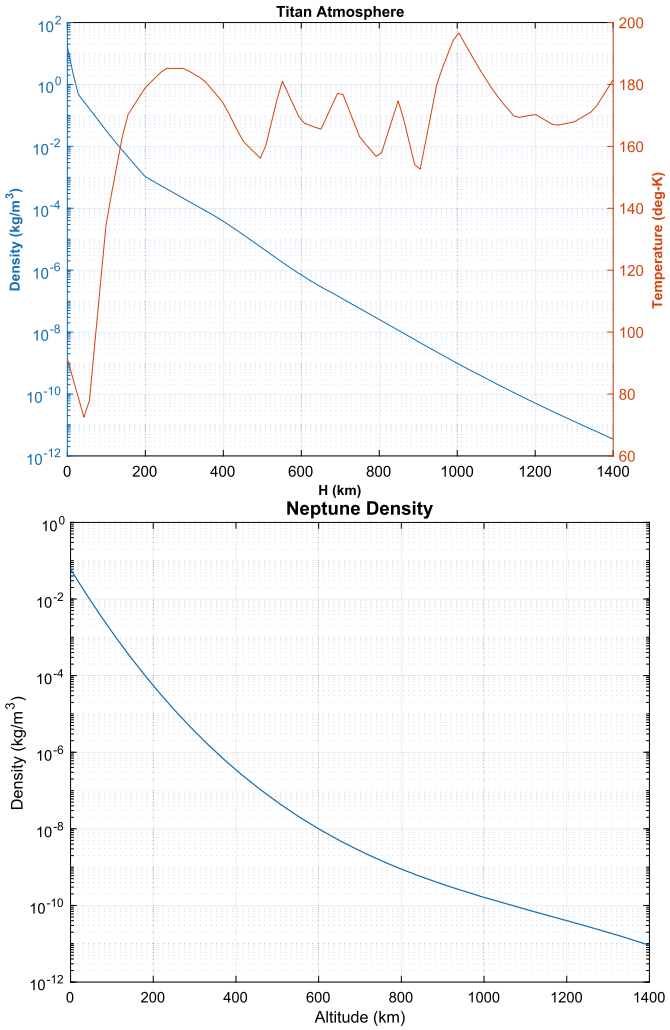


Figure 7.2 Atmospheric properties for Titan and Neptune.

7.6.2 Spherical harmonics

The spherical-harmonic expansion method works as follows. Define the perturbing gravitational potential of a planet as

$$V = \frac{\mu}{r} \sum_{n=2}^{\infty} \left[\left(\frac{a}{r} \right)^n \sum_{m=0}^{\infty} \left(S_{n,m} \sin m\lambda + C_{n,m} \cos m\lambda \right) P_{n,m}(\sin \phi) \right] \tag{7.8}$$

defined in the planet fixed frame. a is the radius of the planet. The definition of the coordinates is shown in Fig. 7.3.

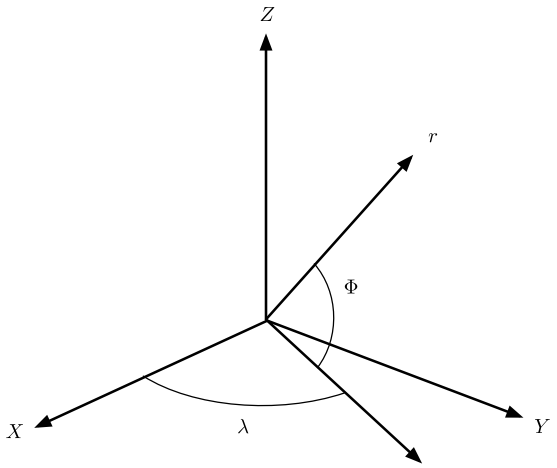


Figure 7.3 Coordinates for a nonhomogeneous planet.

If we define i, j, k as unit vectors along the planetary x-, y-, and z-axes, the gravitational acceleration can be found as follows

$$\frac{\partial V}{\partial r} = \sum_{n=2}^{\infty} \sum_{m=0}^{n} \frac{\partial V_{n,m}}{\partial r} \tag{7.9}$$

and

$$\frac{\partial V_{n,m}}{\partial r} = \frac{\mu}{r^2} \left(\frac{a}{r}\right)^n \left[-r(vH_{n,m} + B_{n,m}) + iD_{n,m} - jE_{n,m} + kH_{n,m}\right] \tag{7.10}$$

$$B_{n,m} = (C_{n,m}\hat{C}_m + S_{n,m}\hat{S}_m)(n+m+1)P_n^m \tag{7.11}$$

$$E_{n,m} = -m(C_{n,m}\hat{S}_{m-1} - S_{n,m}\hat{C}_{m-1})$$

$$D_{n,m} = m(C_{n,m}\hat{C}_{m-1} + S_{n,m}\hat{S}_{m-1})$$

$$H_{n,m} = (C_{n,m}\hat{C}_m + S_{n,m}\hat{S}_m)P_n^{m+1}$$

$$\hat{C}_m = \hat{C}_1\hat{C}_{m-1} - \hat{S}_1\hat{S}_{m-1}$$

$$\hat{S}_m = \hat{S}_1\hat{C}_{m-1} + \hat{C}_1\hat{S}_{m-1}$$

The starting conditions are

$$\begin{aligned} \hat{C}_0 &= 1 \quad \hat{S}_0 = 0 \\ \hat{C}_1 &= \tfrac{x}{r} \quad \hat{S}_1 = \tfrac{y}{r} \end{aligned} \tag{7.12}$$

The derivatives of the Legendre polynomials are

$$P_n^0 = \frac{1}{n}\left[(2n-1)\nu P_{n-1}^0 - (n-1)P_{n-2}^0\right] \tag{7.13}$$

$$P_n^m = P_{n-2}^m + (2n-1)P_{n-1}^{m-1} \tag{7.14}$$

$$P_0^0 = 1$$

$$P_1^0 = \frac{z}{r} \equiv \nu$$

$$P_0^1 = 0$$

$$P_1^1 = 1$$

This kind of harmonic expansion is the standard method for modeling a planet's gravitational field. Harmonic models exist for the Earth, Moon, Mars, and other planets. A harmonic model is convenient because it gives an idea of how much influence higher-order terms have on the overall force.

An Earth 68-by-68 spherical-harmonic gravity model is shown in Fig. 7.4.

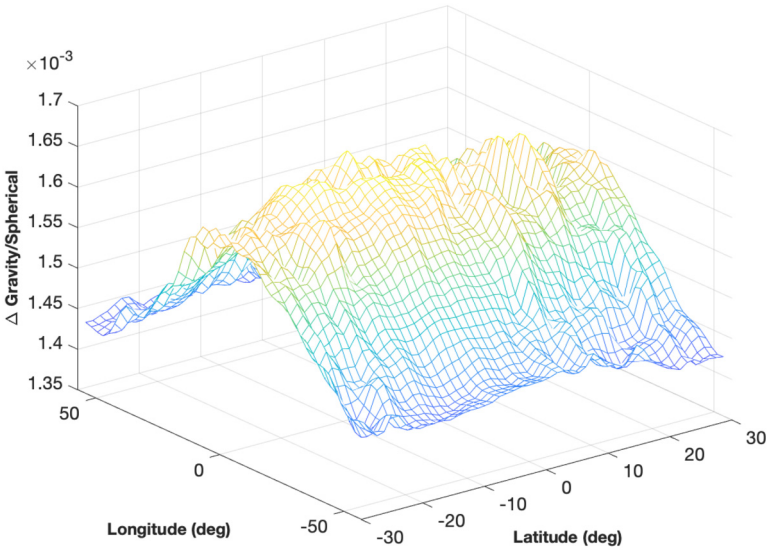


Figure 7.4 Earth 68-by-68 spherical-harmonic gravity model.

The lunar 150-by-150 spherical-harmonic gravity model is shown in Fig. 7.5. Note the "lumps" that are indicative of mass concentrations.

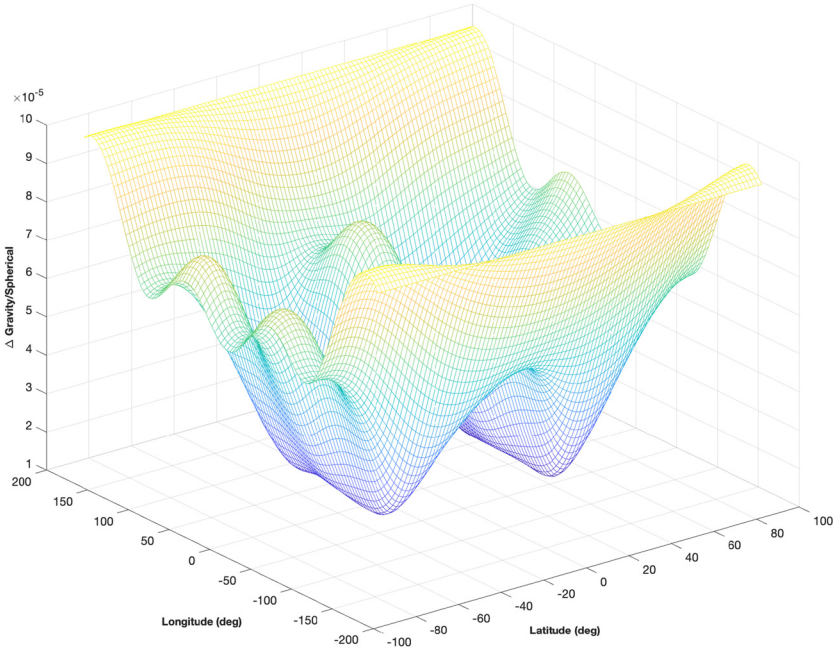


Figure 7.5 150-by-150 spherical-harmonic lunar gravity model. The plot shows magnitude variations.

7.7. Magnetic fields

The following planets and moons have significant magnetic fields

1. Earth;
2. Mars;
3. Jupiter;
4. Saturn;
5. Uranus;
6. Neptune;
7. Ganymede [13].

The simplest magnetic-field model is a tilted dipole model of the magnetic field.

$$g_{10} = g_{10}(0) + \frac{dg_{10}}{dt}(t - t(0)) \tag{7.15}$$

$$g_{11} = g_{11}(0) + \frac{dg_{11}}{dt}(t - t(0)) \tag{7.16}$$

$$h_{11} = h_{11}(0) + \frac{dh_{11}}{dt}(t - t(0)) \tag{7.17}$$

$$h_0 = \sqrt{h_{11}^2 + g_{11}^2 + g_{10}^2} \tag{7.18}$$

$$\theta_M = \cos^{-1}\frac{g_{10}}{h_0} \tag{7.19}$$

$$\phi_M = \tan^{-1}\frac{h_{11}}{g_{11}} \tag{7.20}$$

$$b = \frac{h_0 a^3}{r^3}\begin{bmatrix} \sin\theta_M \cos\phi_M \\ \sin\theta_M \sin\phi_M \\ \cos\theta_M \end{bmatrix} \tag{7.21}$$

where t is time, a = 6371 km, and r is the spacecraft distance from the center of the Earth. This model has a linear variation of coefficients with time. Most tables of magnetic fields include the time coefficients.

The Earth's dipole is shown in Fig. 7.6 in a 55-degree inclined orbit in the ECI frame.

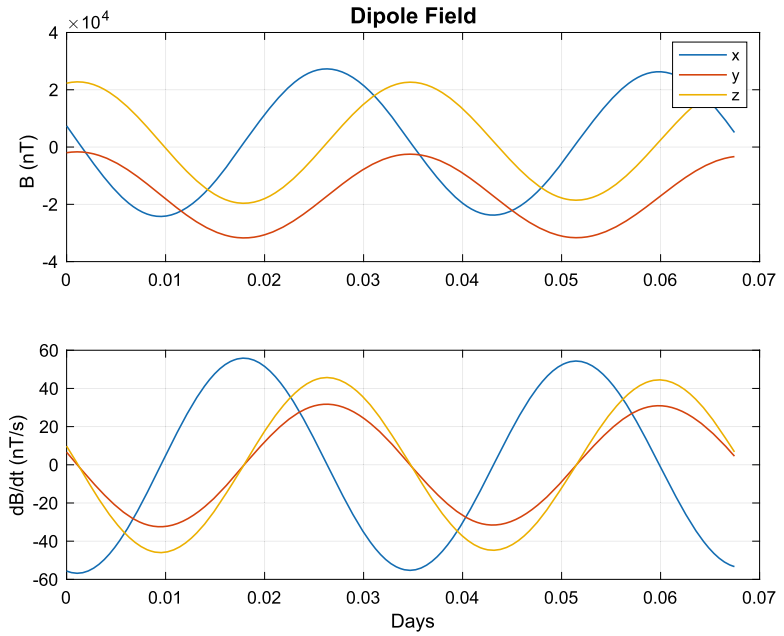
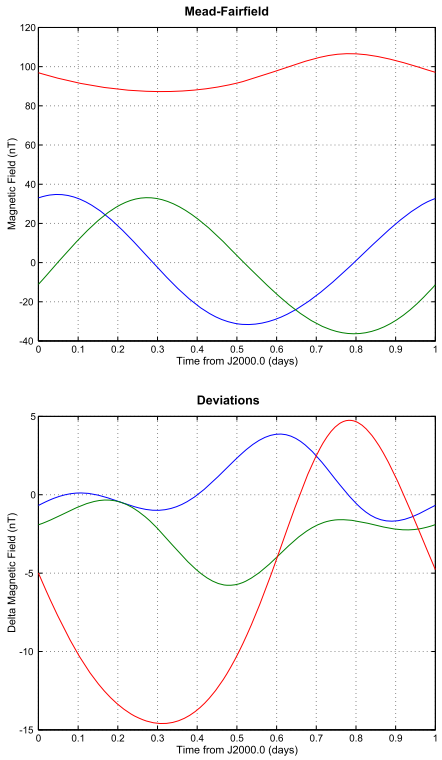


Figure 7.6 The Earth's dipole field in a 55-degree inclined orbit.

The second, suitable for orbits greater than 10 000 km, includes the effect of the Sun and is known as the Mead–Fairfield model [12]. Mead–Fairfield is a quantitative model, based on spacecraft magnetometer data, that perturbs a dipole model to account for the Sun's activity. An input is the solar-activity index. The third is a complete spherical-harmonic expansion of the Earth's magnetic field. Example 7.3 shows the difference between the dipole and Meade–Fairfield model. The Mead–Fairfield model should be used for geosynchronous spacecraft.

```
1 a  = linspace(0,2*pi);
2 r  = 42167*[cos(a);sin(a);zeros(1,100)];
3 jD = JD2000 + linspace(0,1);
4 BMF( r, jD );
```



Example 7.3: Difference between a dipole and Meade–Fairfield magnetic-field model.

The spherical-harmonic expansion is similar to that of gravity models.

7.8. Ionizing radiation

Ionizing radiation can either be cosmic background radiation or ionizing radiation from radiation belts around a planet. The following planets have radiation belts.

1. Earth;
2. Jupiter;
3. Saturn;
4. Uranus;
5. Neptune.

Jupiter is by far the most intense. The others are on the order of the Earth's [13]. Fig. 7.7 shows the Earth's Van Allen belts.

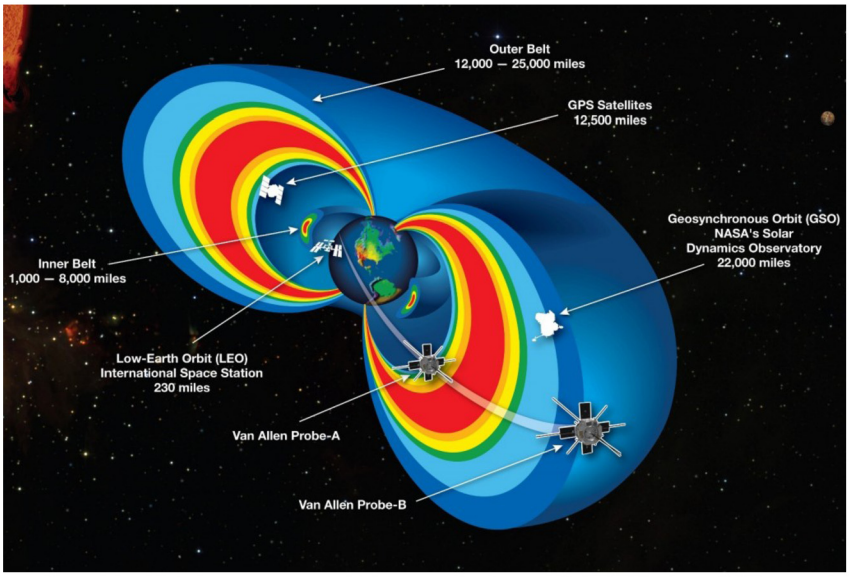


Figure 7.7 The Earth's Van Allen radiation belts. Image Source: http://www.nasa.gov/mission_pages/rbsp/multimedia/20130228_briefing_materials.html.

Fig. 7.8 shows another view with a spacecraft orbit.

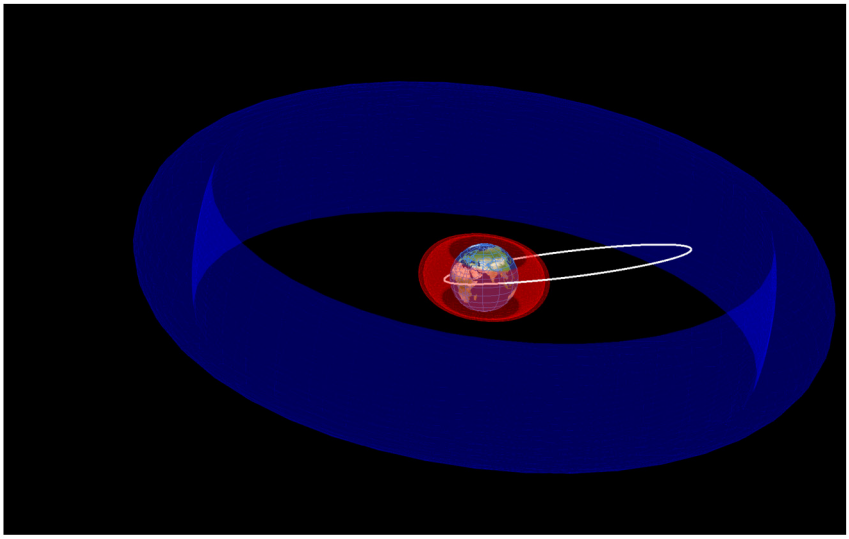


Figure 7.8 Another view showing the inner and outer belts. An inclined orbit is shown for reference.

Ionizing-radiation damage can cause long-term damage to electronics, including solar cells. Ionizing radiation can also have immediate effects:

1. Single-event upsets - when a high-energy particle changes the state of a transistor gate. This is a soft, transient error. It can cause noise in CMOS or CCD chips. A high-energy particle can produce a nuclear reaction at a site. A gate can be destroyed by a high-energy particle.
2. Latch-up - this is a subset of single-event upsets when a particle causes a high current in a transistor that locks up the electronics. This requires cycling the power to clear the problem.

Electronics for attitude control systems should, at the least, be latch-up immune. SEUs for memory are handled by using additional bits to verify the words and by locating the bits on different areas of the memory chip.

Another effect is the degradation of solar cells. The end-of-life power will be lower than the beginning-of-life power. The ACS must be able to operate with lower power levels when the spacecraft is ending its mission life.

References

[1] S.G. Turyshev, V.T. Toth, G. Kinsella, S.-C. Lee, S.M. Lok, J. Ellis, Support for the thermal origin of the pioneer anomaly, Physical Review Letters 108 (Jun 2012) 241101.
[2] D. Hastings, H. Garrett, Spacecraft-Environment Interactions, Cambridge Atmospheric and Space Sciences Series, Cambridge University Press, 1996.
[3] Andrew Marshall, S.B. Luthcke, Radiative force model performance for TOPEX/Poseidon precision orbit determination, The Journal of the Astronautical Sciences 42 (2) (April-June 1994) 229–246.
[4] P.C. Knocke, J.C. Ries, B.D. Tapley, Earth radiation pressure effects on satellites, in: AIAA/AAS Astrodynamics Conference, AIAA, August 1988, pp. 577–583, AIAA-88-4292.
[5] I. Ashdown, Radiosity: A Programmer's Perspective, John Wiley and Sons, 1994.
[6] K. Akins, L. Healy, S. Coffey, M. Picone, Comparison OF MSIS AND Jacchia atmospheric density models for orbit determination and propagation, in: 13th AAS/AIAA Space Flight Mechanics Meeting, American Astronautical Society, 2003, AAS 03-165.
[7] M. Storz, B. Bowman, J. Branson, High accuracy satellite drag model (HASDM), in: 34th COSPAR Scientific Assembly, in: COSPAR, Plenary Meeting, vol. 34, 2002.
[8] M. Storz, B. Bowman, J. Branson, High Accuracy Satellite Drag Model (HASDM), in: AIAA/AAS Astrodynamics Specialist Conference and Exhibit, AIAA/AAS, 2003.
[9] E. Doornbos, H. Klinkrad, Atmospheric density calibration using satellite drag observations, in: Second European Space Weather Week, November 2005.
[10] D. Hastings, H. Garrett, Spacecraft-Environment Interactions, Cambridge Atmospheric and Space Sciences Series, Cambridge University Press, 1996, pp. 142–207.
[11] H.B. Garrett, A.C. Whittlesey, Guide to Mitigating Spacecraft Charging Effects, Tech. Rep., Jet Propulsion Laboratory, June 2011.
[12] G. Mead, D. Fairfield, A quantitative magnetosphere model derived from spacecraft magnetometer data, Journal of Geophysical Research 80 (4) (February 1975) 523–534.
[13] T. Bland, A. Showman, G. Tobie, Production of Ganymede's magnetic field, Icarus 198 (2) (2008) 384–399.

CHAPTER 8

Disturbances

8.1. Space story

I was sitting at my desk when my boss came in and said, "You have to go to Japan now!" It turned out their satellite was losing roll control. I was able to delay a day but then had to fly from Newark to Tokyo and then take a taxi down to the Kimitsu ground station. The magnetic-torquer control system did not have enough control authority because it was undersized for December when the Sun suppressed the Earth's magnetic field. The torquers operated in saturation. Fortunately, each torquer had a backup coil. When that was turned on there was just enough additional torque to control the spacecraft. The problem arose because of a faulty disturbance model that did not account for the Sun's effect. The Sun causes the atmosphere to expand on the opposite side of the Earth. This distorts the magnetic field changing the field seen by the spacecraft.

8.2. Introduction

This chapter discusses torque and force disturbances on a spacecraft. Disturbances may be internal or external. External disturbances must be countered by actuators and are used to size actuators. The ultimate performance of the spacecraft will depend on the magnitude and direction of the disturbances. For those familiar with aircraft the magnitude of the disturbances, often on the order of micro-Newton meters (μN m) seem very small. However, because they act on the spacecraft for a long period, they can cause considerable perturbations in the spacecraft orientation.

External disturbances are a function of the orientation of the spacecraft, its orbital position and velocity, the relative position of nearby planets, and the orientation of the Sun with respect to the spacecraft. Because external disturbances are due to interaction with the spacecraft the type of surfaces is also important. Surfaces can change once the spacecraft is on-orbit so these changes must also be considered. The torque is often a function of the center-of-mass of the spacecraft. Gravity-gradient torques are determined by the inertia matrix.

Internal disturbances can cause pointing errors and may also require either active or passive compensation. Internal disturbances typically come from moving parts in the spacecraft. Sometimes the control actuators, such as reaction wheels, are a source of disturbances. The reaction-wheel noise required a suspension on the Hubble Space Telescope and precluded their use entirely on the Gravity Probe B spacecraft.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00019-X

Disturbances due to outside sources, such as gravity gradient, aerodynamic, and solar, are functions of the attitude or orientation and in the case of aerodynamic disturbances, the velocity of the spacecraft. They cause changes in the dynamical equations of the spacecraft and must be accounted for when modeling the dynamics. Fig. 8.1 shows how disturbance analysis fits into the design process. The column on the left lists the required inputs for modeling disturbances. The surface model is a surface mesh, usually consisting of triangles. There are two broad classes of disturbances, those that are dependent on the surface geometry and those that are independent of the surface geometry. All of the external-force models are a function of the orbit and other celestial objects such as the Sun or nearby planets or moons. Gravity gradient is dependent on the position of the satellite, the gravitational constant of the nearby planet, and the spacecraft inertia matrix. Residual dipole, radio-frequency disturbances, and outgassing depend on spacecraft properties.

The blocks on the left are the information sources for disturbances. The ephemeris is the position of the Sun and other sources of fluxes. Eclipses are a function of the relative position of the Sun and nearby planets or asteroids. Atmosphere models determine the atmospheric density that drives the aerodynamics forces. The magnetic field determines the residual dipole torque and also the torque produced by any magnetic torquers. The inertia determines the gravity-gradient torque. The center-of-mass, combined with forces due to external fluxes, determines the torques due to the external fluxes. The spacecraft hardware determines internal torques and some forces and torques, such as outgassing and thermal.

For complex geometries, shadowing may have to be considered. Solar wings, antennas, and other deployed structures can shadow other parts of the spacecraft. This can have a large impact on the disturbances.

Disturbance modeling is used for two purposes. One is sizing actuators and the other is for performance simulations. The disturbances on the spacecraft are one factor in sizing momentum-exchange devices. The required torque must be larger than the instantaneous disturbance torque and the momentum storage is based on the steady inertial torque. If thrusters are used, the disturbance level will determine the required minimum impulse bit. Other factors, such as required maneuvers, also factor into actuator sizing.

Simulations do not require high-fidelity disturbance models. In some cases, if the spacecraft does not change its orientation, a Fourier-series representation can be used. If the spacecraft maneuvers, the disturbance model needs to be incorporated into the simulation. The model can be of lower fidelity than that used for the disturbance analysis.

This chapter will provide models for all external disturbances. Internal disturbances will be discussed in the context of the actuators in the chapter on actuators.

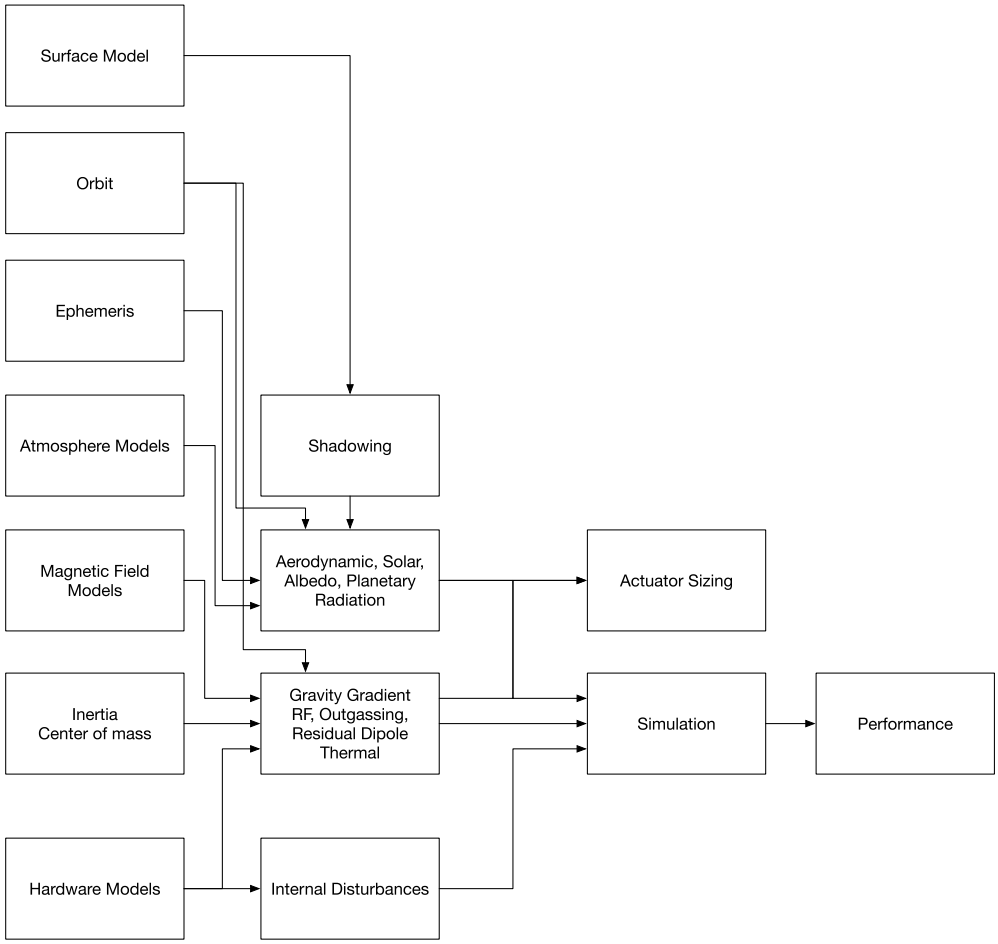


Figure 8.1 Disturbance analysis fits into the spacecraft-design process in several places.

8.3. External disturbances

External disturbances change the angular momentum of the spacecraft. Disturbances can produce periodic or constant momentum changes. The former will cause the spacecraft to oscillate (or if controlled by reaction wheels) the wheel speed to oscillate at harmonics of the orbit rate. The latter will cause an increase in the spacecraft's angular momentum. This increase requires external torquers, typically provided by thrusters or magnetic torquers, to control the increase. It is necessary to tabulate all of the disturbances in a disturbance budget for the spacecraft. The budget, which has temporal categories, is used to size actuators and/or momentum storage. Models of the disturbances are also used in simulations of the spacecraft.

It may be surprising to talk about radiation pressure. In the early days of spaceflight, not everyone was convinced that light could produce a force. Table 8.1 lists major external disturbances on spacecraft. Some of these are discussed in the following sections.

Table 8.1 Disturbances.

Type	Source	Characteristics
Aerodynamic	Planetary atmosphere.	Drag and lift on the spacecraft. A strong function of altitude and position relative to the Sun Most spacecraft do not have lift. The surface of the spacecraft is very important.
Albedo Pressure	Sun reflection from the Earth.	The average is 0.34 of the solar flux for the Earth. Albedo varies depending on the latitude and longitude of the spacecraft, season, and specific surface properties of the planet over which the spacecraft is flying. The relative angle of the Sun determines the direction of the force.
Electrodynamic Force	Motion through Earth's magnetic field.	Produces drag or lift on a satellite. Varies depending on the strength of the magnetic field and the orientation of the satellite's path through the magnetic field. This is a very small effect and is rarely considered in a disturbance analysis.
Gravity gradient	Inertia and distance from the planet.	This is due to the distributed nature of a satellite. If the satellite has off-diagonal inertia terms it will produce a body-fixed torque for a planet-pointing spacecraft. The diagonal terms cause a gravity-gradient modal frequency. A gravity-gradient-stabilized spacecraft uses a specific distribution of momentum to provide a stable oscillation. Gravity gradient is a pure stiffness term in the dynamical model. It cannot provide damping, which must be provided by another source.
Leaks	Onboard gas and liquid supplies.	Acts like a cold gas thruster. Leaks are not deliberate but may happen due to damage to the spacecraft. Pressurized spacecraft with human crews, must account for potential leaks. Waste dumps are a source of forces and torques. Spacecraft with crews are massive so these forces are not usually a major problem.
Outgassing	Moisture embedded in the structure.	Caused by the heating of surfaces and the resulting emission of gas. This often happens with composite solar arrays and can produce large forces and torques. Outgassing is usually only an issue at the beginning of life for spacecraft.

continued on next page

Table 8.1 (*continued*)

Type	Source	Characteristics
Thruster Plumes	Interaction of rocket exhaust plumes and the structure.	Thruster plumes are mathematically the same as aerodynamics forces and torques. They are modeled in two broad regimes, near-field and far-field. Far-field is when a thruster plume impinges on a surface far enough away from the thruster so that the flow is not a continuum. In the near-field, the flow is denser and fluid-dynamic models may be appropriate.
Radiation Pressure	Temperature of the planet.	Black-body radiation of the planet. The value is 400 W for the Earth. This radiation pressure is at infrared wavelengths. When modeling planetary radiation, the optical properties at infrared wavelengths must be used.
Residual Dipole	Residual dipole on the spacecraft.	This torque is caused by the interaction of an external magnetic field and internal dipoles due to current loops, etc. Designers make every effort to eliminate current loops but they are not always avoidable. Magnetic torquers, using a torque rod, always have some residual dipole. Motors have magnetic materials and can also produce a residual dipole.
RF	Transmit antennas.	Radiation pressure. Divide the wattage by the speed of light to get the pressure. Communication satellites, with large offset transmitting antennas, can produce significant torquers on the spacecraft.
Solar Pressure	Solar radiation.	The Sun produces large forces on the spacecraft. If the spacecraft is asymmetric, the offset of the center-of-mass and center-of-pressure will result in a large torque. This is why communication satellites usually have symmetric solar wings. The surface force depends on the surface properties. Some spacecraft, such as solar sails, take advantage of this force for propulsion. The solar flux is 1367 W at the Earth orbit and varies inversely with the square of the distance from the Sun.
Thermal Pressure	Radiators.	Heat radiates diffusely from all radiators. This produces a force proportional to the heat flux. The temperature of the spacecraft is not uniform. Some components, such as batteries, may have their own radiators.

External disturbances either produce a force, which produces a torque due to a moment arm with respect to the center-of-mass, or produce a pure torque. This is shown in Fig. 8.2.

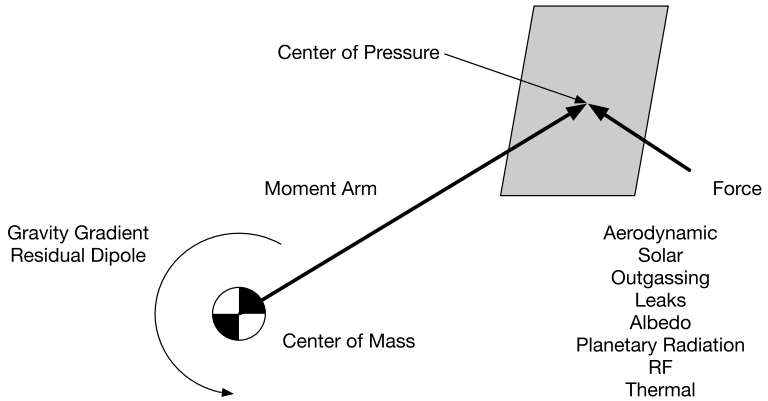


Figure 8.2 Torque production. Some disturbances produce torques without a force.

8.3.1 Surface geometry

The surface mesh should be the simplest possible model that represents the surface. The resolution of the mesh should be such that the error due to mesh resolution is about the same order of magnitude as that of the other parameters in the model. Most spacecraft computer-aided design (CAD) packages generate hundreds of thousands of triangles and provide too much detail to be of use. An important consideration is to only include triangles that are visible to the incoming particle flux or radiation flux. Each surface element is associated with a normal that can be used to eliminate triangles that are not visible. However, care must be taken not to include internal surfaces. Fig. 8.3 shows two objects with front and back surfaces. One is blocked by the other. The normals for Object 1 can be used to select the faces of that object for disturbance calculation. The front normal of Object 2 cannot because it is shadowed.

8.3.2 Aerodynamic

8.3.2.1 Simple drag coefficient

Aerodynamic disturbances are due to the interaction between a planetary atmosphere and a spacecraft's surface. It is assumed that the interaction is by a stream of particles that do not interact with each other. The simplest model for the aerodynamic force is

$$F = \frac{1}{2}\rho C_D A_p v|v| \tag{8.1}$$

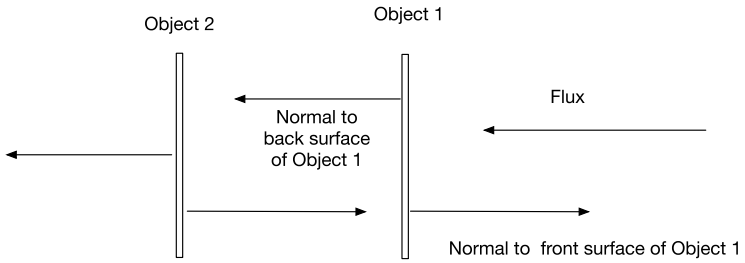


Figure 8.3 Normals can be used to select surfaces in some cases.

where ρ is the atmospheric density, C_D is the drag coefficient, A_p is the projected area, and v is the velocity vector. Lift is perpendicular to the velocity vector. This is the dynamic pressure multiplied by $C_D A_p$. The projected area for a flat plate is

$$A_p = A\cos\alpha \tag{8.2}$$

where α is the angle between the surface normal and the velocity vector. Unlike an aircraft, only surfaces that interact with the particles that arrive with an angle between the velocity vector and the surface normal are considered.

This model assumes that the incoming particle is either reflected directly backward or absorbed by the surface. If it is reflected backward C_D is 4. If it is absorbed, C_D is 2. A is the projected area. If the flow is tangential to the surface, the drag is zero. Determining C_D is difficult. Estimates can be made from orbit-decay data, but this assumes the orientation is known and only measures ρC_D at best.

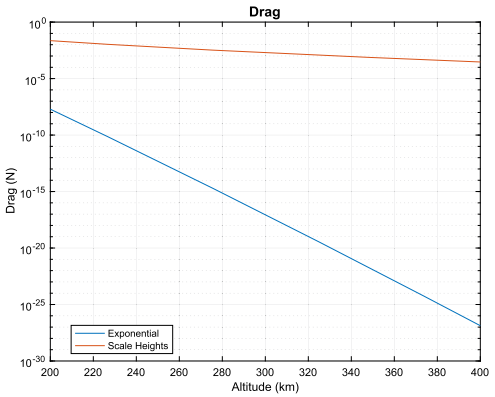
It is useful to look at the magnitude of the forces. Assume the spacecraft has a surface area of 1 m^2 and that the velocity vector is normal to the surface. Example 8.1 shows the drag on a satellite as a function of altitude for different atmosphere models. The exponential atmosphere, which is very popular because of its simplicity, produces a much lower drag force. Scale heights, essentially fit measurements of the atmosphere each over a restricted range, produce large and likely more realistic values. The example shows the problem that the atmosphere poses when modeling drag and computing aerodynamic disturbances.

8.3.2.2 Surface-accommodation coefficients

At altitudes above 120 km, the atmospheric density is sufficiently low that continuum flow does not apply. The Knudsen number shows the flow regime; less than 0.01 is continuum flow, greater than 10 is a free-molecular flow

$$Kn = \frac{\lambda}{L} \tag{8.3}$$

```
% Drag
h      = linspace(200,400);
rE     = 6378.165;
rho1   = AtmDens1(h);
rho2   = AtmDens2(h);
v      = 1000*sqrt(Constant('mu earth')./(rE + h))
        ;
cD     = 2.7;
a      = 1;
d      = 0.5*cD*a*[rho1;rho2].*v.^2;
Plot2D(h,d,'Altitude (km)','Drag (N)','Drag',...
   'ylog',[],[],[],[],[],{{'Exponential' 'Scale
        Heights'}});
```



Example 8.1: Drag for different atmosphere models.

where λ is the mean free path and L is a characteristic dimension of the spacecraft. L for a 1U CubeSat is 10 cm. For the International Space Station (ISS) it is 73 m to 109 m.

The mean free path for a Boltzmann gas is

$$\lambda = \frac{\mu}{\rho}\sqrt{\frac{\pi m}{2k_B T}} \tag{8.4}$$

where μ is the dynamic viscosity, k_B is the Boltzmann constant $= 1.38 \times 10^{-23}$ J/K, m is the molecular weight, and T is the temperature. Kn becomes

$$Kn = \frac{\mu}{\rho L}\sqrt{\frac{\pi m}{2k_B T}} \tag{8.5}$$

An alternative formulation is

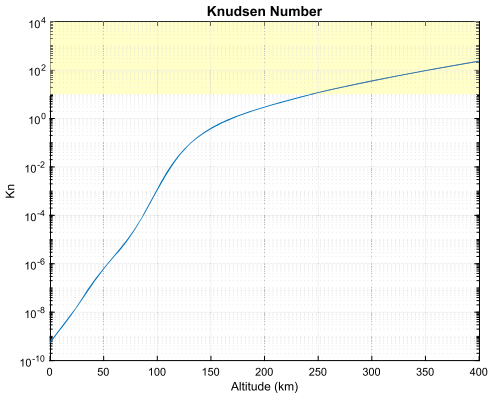
$$Kn = \frac{k_B T}{\sqrt{2}\pi d^2 p L} \tag{8.6}$$

where d is the hard-shell diameter of the particles and p is the total pressure. The hard-shell diameters of some molecules are given Table 8.2. Example 8.2 shows the Knudsen numbers starting at sea level. The transition to free-molecular flow is at 250 km.

Table 8.2 Hard-shell or kinetic diameters.

Name	Formula	Molecular Weight	Diameter (pm)
Hydrogen	H_2	2	289
Water	H_2O	18	265
Nitrogen	N_2	28	364
Carbon Dioxide	CO_2	44	330

```
h    = linspace(0,400);
rho  = AtmDens2(h);
T    = AtmTemp(h);
m    = 28*1.66053886e-27;
kB   = 1.38e-23;
mu   = AbsoluteViscosity( 'n2', T );
l    = 109;
kN   = (mu./(rho*l)).*sqrt(pi*m./(2*kB*T));
Plot2D(h,kN,'Altitude (km)', 'Kn', 'Knudsen
    Number','ylog');
x    = get(gca,'xlim');
y    = get(gca,'ylim');
hold on
yL   = 10;
fill([x(1) x(1) x(2) x(2)],[yL y(2) y(2) yL],[1 1
    0],'edgecolor','none','facealpha',0.2);
```



Example 8.2: Knudsen number. The shaded region is the free-molecular regime.

Free-molecular flow is shown in Fig. 8.4. An incoming particle strikes the surface. It can be absorbed, reflected specularly or reflected diffusely.

Surface-accommodation coefficients [1,2] incorporate these effects.

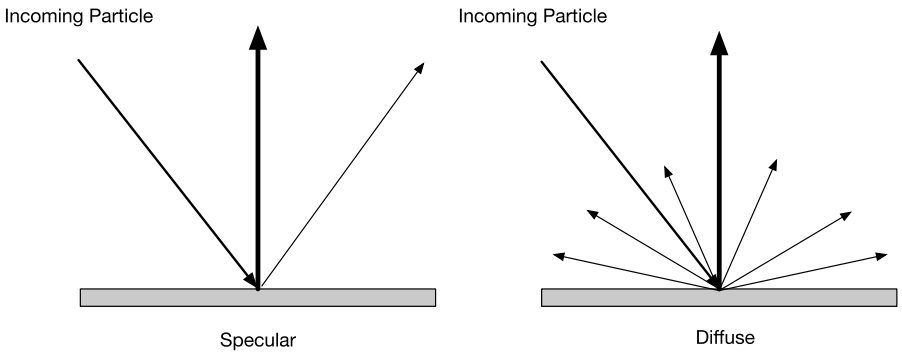


Figure 8.4 Free molecular flow.

The surface-accommodation coefficients are

$$\sigma_n = \frac{p_i - p_r}{p_i - p_w} \tag{8.7}$$

$$\sigma_t = \frac{\tau_i - \tau_r}{\tau_i - \tau_w} \tag{8.8}$$

where p_i and p_r are the incident and reflected normal momentum flux, τ_i and τ_r are the incident and reflected tangential momentum flux, p_w and τ_w are the normal and tangential components of momentum that would be carried away by diffusively emitted momentum, and τ_w is by definition zero. For purely specular reflection with, $p_i = p_r$ and $\tau_i = \tau_r$, $\sigma_n = \sigma_t = 0$.

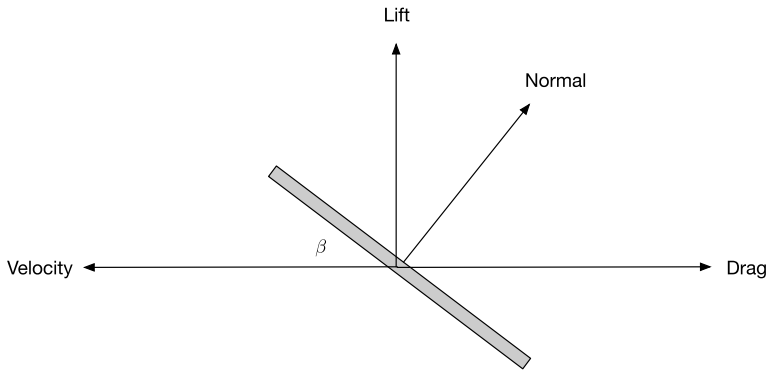


Figure 8.5 Lift and drag on a plate.

There are two approaches to computing the aerodynamic forces. One assumes that the only contribution is the motion of the spacecraft. The second assumes that the particles in the atmosphere have a Maxwellian velocity distribution.

The average normal velocity of diffusely reflected molecules is

$$V_w = \sqrt{\frac{\pi R T_w}{2\mathfrak{M}}} \tag{8.9}$$

where R is the ideal gas constant, which is 8314.5 joules/kg-mole K, $\mathfrak{M}$ is the mean molecular weight, and T is the gas temperature that can be found from the atmosphere model. The surface-accommodation coefficients are also a function of the incidence angle. Results obtained by bombarding an aluminum surface with nitrogen are [1]

$$\sigma_n = 0.93 - 1.48 \times 10^{-3}\theta - 7 \times 10^{-5}\theta^2 \tag{8.10}$$

$$\sigma_t = 0.63(1 - e^{-3.38\times 10^{-2}\theta}) \tag{8.11}$$

Example 8.3 shows the normal and tangential accommodation coefficients.

Since the surface in the experiment was not molecularly clean, and all spacecraft surfaces are coated with a layer of monatomic oxygen, this result applies to most spacecraft surfaces.

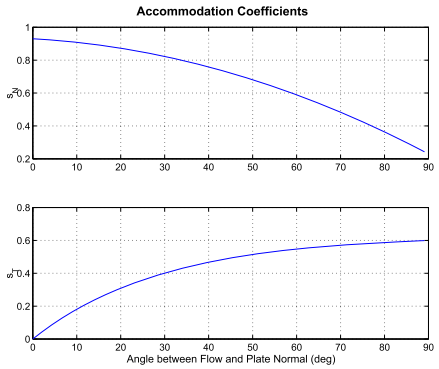
Fig. 8.5 shows the lift and drag geometry. For the zero atmospheric-motion case, we get

$$C_L = \left[\sigma_n \frac{V_w}{V} + (2 - \sigma_n - \sigma_t)\sin\beta\right]\sin 2\beta \tag{8.12}$$

$$C_D = 2\left[\sigma_t + \sigma_n \frac{V_w}{V}\sin\beta + (2 - \sigma_n - \sigma_t)\sin^2\beta\right]\sin\beta \tag{8.13}$$

For the specular case with $\beta = \pi/2$, $C_D = 4$.

```
1 FAeroSurfaceAccommodation
```



Example 8.3: Surface-accommodation coefficients.

With a Maxwellian velocity distribution, the equations are more complex

$$C_L = \left[\frac{2-\sigma_n-\sigma_t}{S\sqrt{\pi}} e^{-S^2\sin^2\beta} + \sigma_n \frac{V_w}{V}\right] \sin 2\beta \tag{8.14}$$

$$+ \left[\frac{2-\sigma_n}{S^2} + 2(2-\sigma_n-\sigma_t)\sin^2\beta\right] \cos\beta \ \text{erf}(S\sin\beta) \tag{8.15}$$

$$C_D = 2\sigma_n \frac{V_w}{V}\sin^2\beta + \frac{2}{\sqrt{\pi}S^2}\left[(2-\sigma_n)\sin^2\beta + \sigma_t\cos^2\beta\right] e^{-S^2\sin^2\beta} \tag{8.16}$$

$$+ 2\left[(2-\sigma_n)\left(\sin^2\beta + \frac{1}{2S^2}\right) + \sigma_t\cos^2\beta\right] \sin\beta \ \text{erf}(S\sin\beta) \tag{8.17}$$

where

$$S = \frac{V}{V_a} \tag{8.18}$$

and average velocity of the particles in the atmosphere is

$$V_a = \sqrt{\frac{\pi R T_a}{2\mathfrak{M}}} \tag{8.19}$$

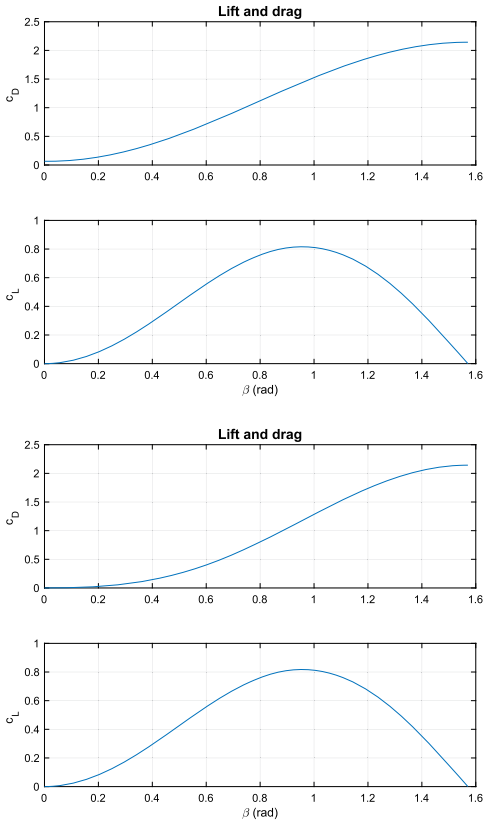
where T_a is the atmosphere temperature.

Example 8.4 shows the two models. The results are similar.

8.3.3 Electrodynamic force

An electrodynamic force is the result of the Lorentz force that occurs when a current-bearing wire moves through a magnetic field. In the context of satellite disturbances, the magnetic field is the geomagnetic field. This force can be exploited to produce either drag or lift for a satellite by an electrodynamic tether.

```
h      = 350; % km
m      = 28;
tA     = AtmTemp(h);
tW     = 300; % deg-K
v      = sqrt(3.986e5/(6378+h))*1000;
beta   = linspace(0,pi/2);
CLCDFreeMolecular( v, m, tW, beta )
CLCDFreeMolecular( v, m, tW, beta, tA )
```



Example 8.4: Free-molecular flow drag and lift coefficients. The bottom plot includes the Maxwellian atmospheric velocity.

Satellites attached to electrodynamic tethers experience electrodynamic forces either aligned with or opposing their motion perpendicular to the magnetic field. The force on the satellite is

$$\mathbf{F} = I \int_0^l d\mathbf{l} \times \mathbf{B} \tag{8.20}$$

where I is the current in the wire and $\mathbf{B}$ is the magnetic field. The current flowing through the tether will have a natural value that is a result of moving through the magnetic field that can be derived from Ohm's law,

$$I = \Phi / R \tag{8.21}$$

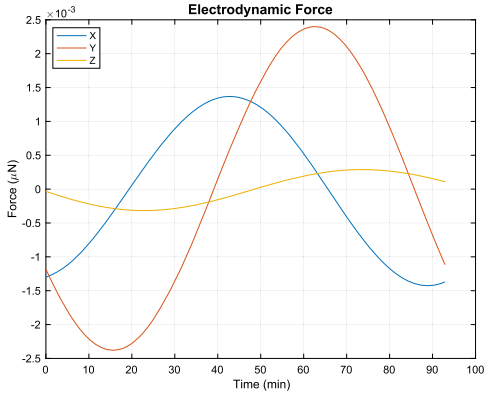
where Φ is the voltage produced across the tether and R is the total resistance of the tether. For a satellite moving through the geomagnetic field with velocity $\mathbf{v}$, the voltage

induced in the tether is given by

$$\Phi = \int_0^l \mathbf{v} \times \mathbf{B}\, d\mathbf{l} \tag{8.22}$$

The natural current produces a force that resists the motion of the satellite and has the effect of a drag force on the satellite's orbit. Alternatively, the current intensity in the tether can be controlled by applying an external voltage to the system. A current will also produce an electromagnetic force shown in (8.20) and can be used to boost a satellite's orbit. Example 8.5 shows the electrodynamic force on a 100-m tether aligned with nadir in an ISS orbit. The tether is American Wire Gauge (AWG) 10 copper wire. We approximate the velocity and magnetic field as being constant across the length of the tether.

```
[el,jD0]  = ISSOrbit;
[r,v,t]   = RVOrbGen(el);
jD        = jD0 + t/86400;
b         = BDipole(r,jD);
l         = 100; % m
R         = 3.1932e-03*l; % AWG 10 wire
phi       = l*Cross(v,b);
i         = phi/R;
f         = l*Cross(Unit(r),b);
[t,tL]    = TimeLabl(t);
lL        = {{'X' 'Y' 'Z'}};
Plot2D(t,f,tL,'Force (\muN)','Electrodynamic
      Force',[],[],[],[],[],lL);
```



Example 8.5: Electrodynamic force in an ISS orbit.

8.3.4 Gravity gradient

The gravity-gradient disturbance torque arises from the off-diagonal terms in the inertia matrix. The vector torque is

$$T = 3\mu \frac{r^{\times} Ir}{|r|^5} \tag{8.23}$$

where r is the vector to the satellite from the center of the Earth measured in the spacecraft body frame. This is a general equation and applies to any situation.

It is easiest to understand this torque by working in the Earth-pointing frame, local vertical/local horizontal (LVLH) that rotates at orbit rate. The frame is shown in Fig. 8.6.

The orbit rate ω_o is

$$\omega_o = \sqrt{\frac{\mu}{|r|^3}} \tag{8.24}$$

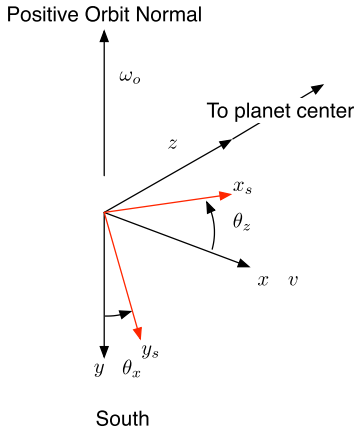


Figure 8.6 LVLH frame for the gravity-gradient analysis.

Assume only small rotations from the LVLH frame. Dividing through by the magnitude of r we get

$$T = 3\omega_o^2 u^{\times} Iu \tag{8.25}$$

where the unit vector is

$$u = \frac{r}{|r|} \tag{8.26}$$

The r vector is along the LVLH z-axis. The skew-symmetric matrix is

$$u^{\times} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \tag{8.27}$$

The unit vector in the body frame is

$$u = \begin{bmatrix} -\theta_y \\ \theta_x \\ -1 \end{bmatrix} \tag{8.28}$$

This is a "unit" vector if the angles are small. Expanding Eq. (8.25) we get

$$T = 3\omega_o^2 \begin{bmatrix} 0 & -1 & \theta_x \\ 1 & 0 & -\theta_y \\ -\theta_x & -\theta_y & 0 \end{bmatrix} \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix} \begin{bmatrix} -\theta_y \\ \theta_x \\ -1 \end{bmatrix} \tag{8.29}$$

Dropping nonlinear terms we get

$$T = 3\omega_o^2 \left(\begin{bmatrix} I_{zz} - I_{yy} & I_{xy} \\ I_{xy} & I_{zz} - I_{xx} \\ I_{xz} & I_{yz} \end{bmatrix} \begin{bmatrix} \theta_x \\ \theta_y \end{bmatrix} + \begin{bmatrix} I_{yz} \\ -I_{xz} \\ 0 \end{bmatrix} \right) \tag{8.30}$$

The gravity-gradient torque can provide stiffness, depending on the magnitudes of the diagonal terms. Note that these terms need to be combined with the Euler coupling terms to compute the complete gravity-gradient stiffness. The off-diagonal terms produce disturbance torques.

8.3.5 Residual dipole

The residual-dipole disturbances come from the interaction of the magnetic fields generated by current loops on the spacecraft and solar arrays with the Earth's magnetic field. The solar arrays and core spacecraft must be accounted for separately.

The simplest magnetic-field model is a tilted dipole model of the Earth's magnetic field. The second, suitable for orbits greater than 10 000 km, includes the effect of the Sun and is known as the Mead-Fairfield model [3]. Mead–Fairfield is a quantitative model, based on spacecraft magnetometer data, that perturbs a dipole model to account for the Sun activity. Input is the solar activity index. The third is a complete spherical harmonic expansion of the Earth's magnetic field. Example 8.6 shows the difference between the dipole and Meade–Fairfield model. The Mead-Fairfield model should be used for geosynchronous spacecraft.

The torque due to a residual dipole is

$$T = \left(\sum C_k M_k \right) \times B \tag{8.31}$$

where C_k transforms from the frame in which the dipole M_k is expressed to the core body frame and B is the Earth's magnetic field resolved into the body frame. Example 8.7 shows the case for a geosynchronous spacecraft with a Sun-pointing solar array and an Earth-pointing bus each with a 5 ATM^2 residual dipole along the x-axis.

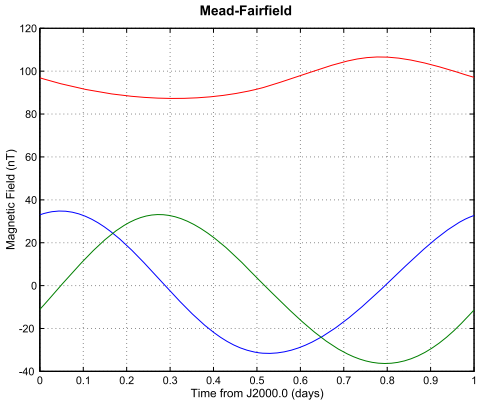
Not all planets have magnetic fields or have very weak fields. Venus has a very weak field, for example. Jupiter, on the other hand, has a very large, intense, and complex field.

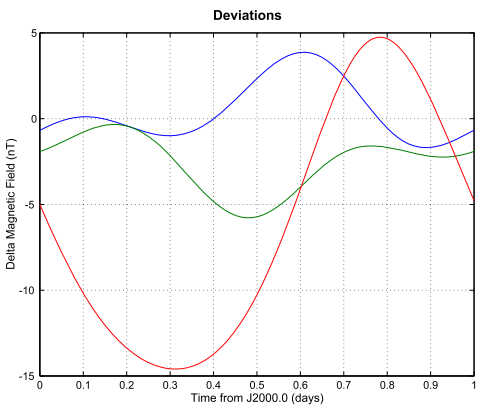
8.3.6 Radio-frequency forces

Transmitting antennas will produce torques. These are known as radio-frequency or RF torques. Any electromagnetic transmission will produce a force and a torque. The torque on the spacecraft will be

$$T = -\frac{p}{c} r^{\times} u \tag{8.32}$$

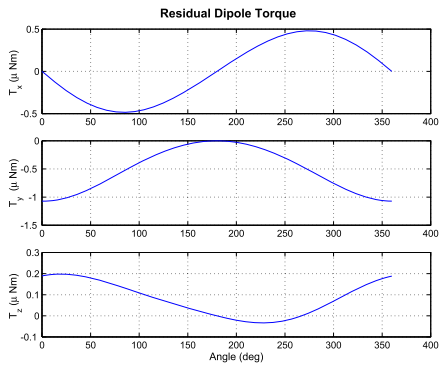
```
1 % Mead-Fairfield model of magnetic field
2 BMF
```





Example 8.6: Difference between a dipole and Meade–Fairfield magnetic-field model.

```
1  a   = linspace(0,2*pi);
2  r   = 42167*[cos(a);sin(a);zeros(1,100)];
3  jD  = Date2JD( [2010 3 21 0 0 0] ) + linspace(0,1)
         ;
4  bT  = BMF( r, jD );
5  n   = length(a);
6  t   = zeros(3,n);
7  d   = [5;0;0]; % Dipole
8  for k = 1:n
9    c       = cos(a(k));
10   s       = sin(a(k));
11   m       = [c s 0;-s c 0;0 0 1];
12   t(:,k) = Cross(d,m*bT(:,k)) + m*Cross(d,bT(:,k)
           );
13 end
14
15 Plot2D( a*180/pi, t*1e6, 'Angle (deg)',...
16         {'T_x (\mu Nm)' 'T_y (\mu Nm)' 'T_z (\mu
                Nm)'},...
17       'Residual Dipole Torque')
```



Example 8.7: Torque due to a residual dipole in geosynchronous orbit.

where p is the transmitted power, u is in the direction opposite the antenna boresight, r is the vector to the antenna boresight from the spacecraft center-of-mass, and c is the speed of light.

Internal RF transmissions, such as between the feeds and reflectors or in waveguides, do not produce a net torque on the spacecraft. Generally, RF poses a problem with high-power satellites with offset reflectors. For example, a transmitting antenna with an output of 600 W offset 3 m from the spacecraft center-of-mass will generate a 6-μN m torque. Since this is a fixed torque over a week a spacecraft will accumulate 3.6 N m s from this disturbance alone.

8.3.7 Solar pressure

Solar pressure is the dominant disturbance on a spacecraft in geosynchronous and interplanetary orbit. Solar pressure is due to the force of photons on the surfaces of the spacecraft. The pressure is

$$s = \frac{p}{c} \tag{8.33}$$

where p is the power of the sunlight and c is the speed of light. As can be seen, a 1-N force would require 3×10^8 W. Thus a flashlight propelled spacecraft would take a long time to gain any velocity.

A photon striking a surface can do one of four things: it can be absorbed, it can reflect specularly (meaning at the same angle with respect to the surface normal that it hit), it can reflect diffusely (meaning at any angle), or it can pass through the surface. Photons that are absorbed must either be transferred somewhere else (through heat conduction or as electricity in the case of a solar array) or be reemitted locally. If the latter happens, the photon must be lumped in with the diffusely reemitted photons. In terms of fractions of the incoming photons, the following is true for a surface

$$1 = \rho_a + \rho_s + \rho_d + \rho_t \tag{8.34}$$

where ρ stands for the fraction of photons that are absorbed, specularly reflected, diffusely reflected, or transmitted.

The solar pressure force is

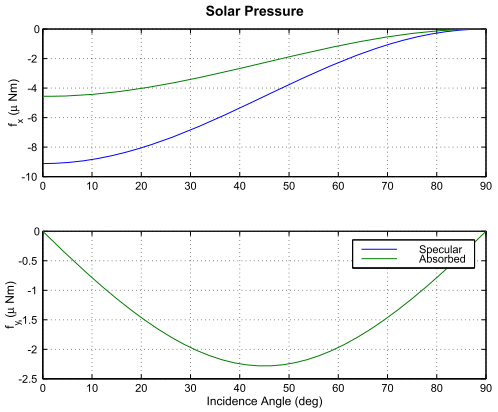
$$F = -SAs^T n(2(\rho_s s^T n + \rho_d/3)n + (\rho_a + \rho_d)s) \tag{8.35}$$

where s is the Sun vector, n is the unit normal to the surface, A is the area of the surface, and S is the solar flux in N/m^2. If the solar array tracks the Sun this becomes

$$F = -SA(\rho_a + \frac{4}{3}\rho_d + 2\rho_s)s \tag{8.36}$$

The specular component produces the biggest force, they diffuse the second largest and the absorbed produces the smallest force contribution. A 1-m^2 surface produces the force shown in Example 8.8 in the cases of pure specular reflection and pure absorption.

```
a = linspace(0,pi/2);
n = length(a);
u = [cos(a);sin(a);zeros(1,n)];
p = 1367/3e8;

fS = zeros(3,n);
fA = zeros(3,n);

for k = 1:n
  fS(:,k) = SolarF( p, [0;1;0;0], [1;0;0], u(:,k)
       , 1 )*1e6;
  fA(:,k) = SolarF( p, [1;0;0;0], [1;0;0], u(:,k)
       , 1 )*1e6;
end

yL = {'f_x (\mu Nm)' 'f_y (\mu Nm)' 'f_z (\mu Nm)
     '};
Plot2D( a*180/pi, [fS;fA], 'Incidence Angle (deg)
     ',...
        yL, 'Solar Pressure',...
        'lin', {'[1 4]' '[2 5]'} )
legend('Specular', 'Absorbed')
```



Example 8.8: Comparison between pure specular and pure absorption.

8.3.8 Earth-albedo force and torque

Albedo forces and torques are due to the reflection of sunlight off the surface and clouds of the Earth. Because the Earth cannot be treated as a point source (as can the Sun) and the reflectivity varies over the surface, albedo is difficult to model accurately [4]. The simplest model is to assume that reflection from the Earth's surface and clouds is diffuse [5]. The flux from the total reflectance can then be computed by integrating over the surface of the Earth.

The Earth's surface is modeled as a Lambertian reflector, as shown in Fig. 8.9.

The first step is to partition the Earth with a cloud map into triangles with each triangle being assigned a diffuse reflection coefficient. This is illustrated on the left of Fig. 8.7. A portion of the radiation reflected from the Earth or clouds intersects the surfaces of the spacecraft as shown on the right of Fig. 8.7. If the spacecraft is more than five times the distance from the surface as the dimensions of the surface, then the following equation can be used to model, the flux [6]

$$\Phi_{ij} = \alpha PA \frac{\cos\theta_i \cos\theta_j}{\pi r^2} \tag{8.37}$$

where α is the absorption coefficient for the thermal flux, P is the solar flux, A is the area of the surface patch, $\cos\theta_i$ is the angle between the normal of the surface patch and the incoming flux, and $\cos\theta_j$ is the angle between the normal of the spacecraft patch

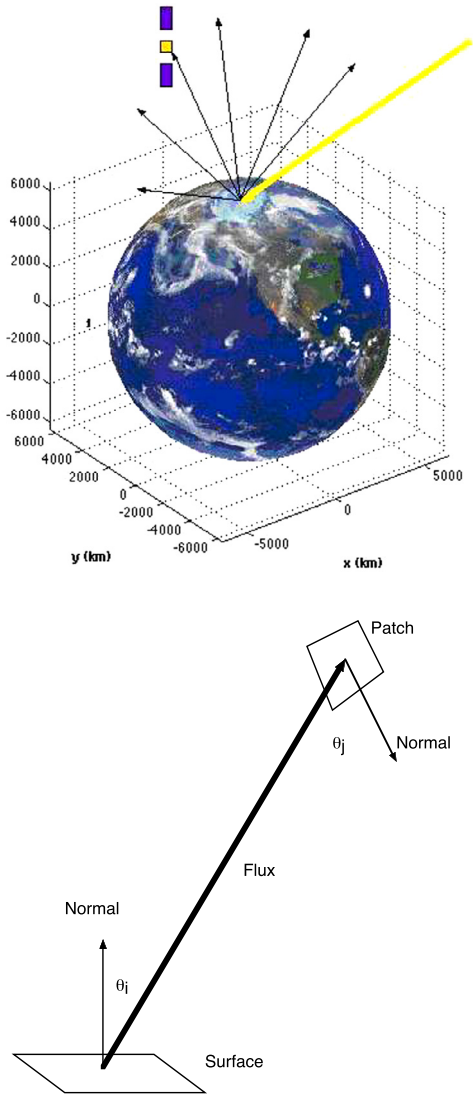


Figure 8.7 Earth albedo and radiosity.

and the incoming radiation. For a spacecraft in a 200-km orbit, this would mean that the triangles representing the surface of the Earth could be no more than 40 km across.

This requires an active subdivision of the Earth's surface so that only the triangles under the spacecraft are tessellated. This is done using repeated tessellation of an icosahedron. An example is shown in Fig. 8.8. The α coefficient varies with each patch and may represent ice, water, vegetation, clouds, etc. The spot under the spacecraft is sub-

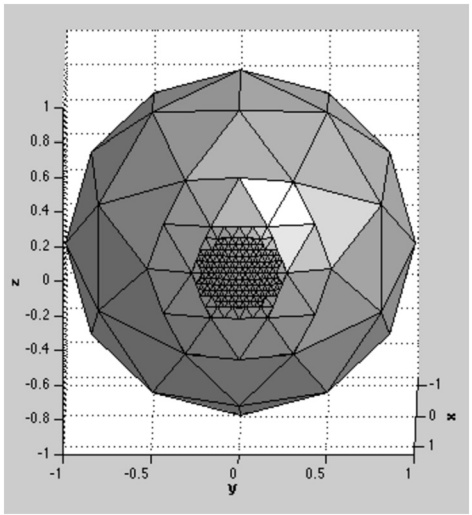


Figure 8.8 Adaptive subdivision.

divided into smaller triangles. The size of the triangles decreases with angular distance from the spacecraft vector.

8.3.9 Planetary-radiation force and torque

Planetary radiation is assumed uniform over the surface of the Earth. The flux at radius r is

$$q = \frac{W}{\left(\frac{r}{r_p}\right)^2} \tag{8.38}$$

where r_p is the radius of the planet and W is the radiation at the surface, which is 400 W for the Earth.

8.3.10 Thermal torque

The radiation emitted from a flat plate is

$$W = \sigma \epsilon A T^4 \tag{8.39}$$

where σ is the Boltzmann constant = 5.67×10^{-12} W/cm^2 K^4, ϵ is the emissivity of the plate, A is the area, and T is the temperature in Kelvin. The radiation is emitted with a distribution proportional to $\cos\phi$, where ϕ is the angle from the normal as illustrated in

Fig. 8.7. Therefore the force coefficient will be

$$\frac{\int_0^{2\pi}\int_0^{\pi/2}\cos^2\phi\sin\phi\,d\phi\,d\theta}{\int_0^{2\pi}\int_0^{\pi/2}\cos\phi\sin\phi\,d\phi\,d\theta}=\frac{\int_0^{\pi/2}\cos^2\phi\sin\phi\,d\phi}{\int_0^{\pi/2}\cos\phi\sin\phi\,d\phi}=\frac{\frac{\cos^3\phi}{3}\Big|_0^{\pi/2}}{\frac{\cos^2\phi}{4}\Big|_0^{\pi/2}}=\frac{2}{3} \tag{8.40}$$

Therefore the force will be

$$F=\frac{2}{3}\frac{\sigma\epsilon AT^4}{c} \tag{8.41}$$

All surfaces will experience a force due to radiative emissions. In steady state, the surface temperature will be what balances radiative input with total output.

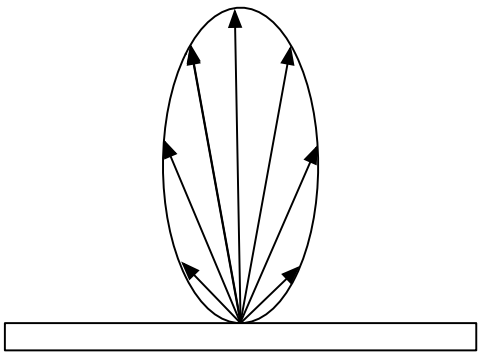


Figure 8.9 Diffuse surface radiation.

If we define the thermal absorptivity of a surface as α, then the steady-state thermal balance is

$$\alpha W=\sigma\epsilon AT^4+q \tag{8.42}$$

per unit area, where W is the incoming solar flux and q is the input or output from the rest of the spacecraft. The remainder of the flux, $1-\alpha$, is either transmitted through the surface or reflected and it may be reflected either specularly or diffusely. The thermal force becomes

$$F=\frac{2}{3}\frac{A}{c}(\alpha W+q) \tag{8.43}$$

8.3.11 Thruster plumes

Thruster plumes cause disturbance torques and forces on the spacecraft when they impact spacecraft surfaces. Plume models are of two types, near-field and far-field. The former apply near the thruster nozzle when the exhaust gas can still be modeled as a

fluid. The latter model is applicable when the gas is best modeled as individual particles. The plume geometry is illustrated in Fig. 8.10.

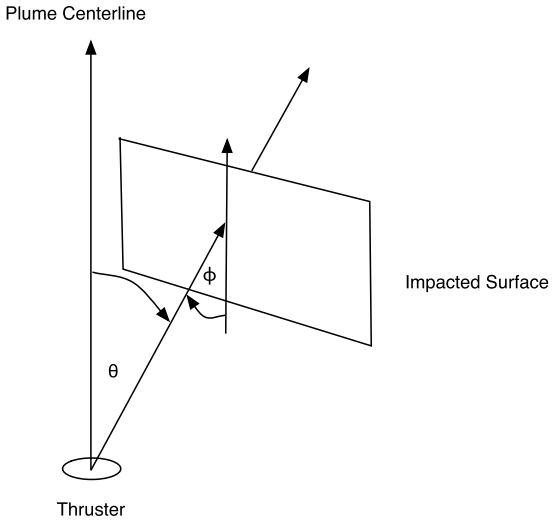


Figure 8.10 Plume-geometry diagram.

The plume is modeled as a far-field disturbance using the method of characteristics to model the plume-flow field. The plume disturbance is assumed to be due solely to interaction with the solar array. The solar array is broken up into panels and the torque is computed at the center of each panel. The plume torque is

$$T = Ar^{\times}p \tag{8.44}$$

where A is the area of the panel, r is the vector from the spacecraft center-of-mass to the panel center-of-pressure, and p is the average plume pressure on the panel. The pressure vector is along the line from the thruster center to the panel center-of-pressure. The plume pressure has a component normal and a component tangential to the surface. The normal pressure component is along the panel antinormal (with the normal defined as pointing towards the thruster.) The tangential component is along the unit vector

$$\frac{n \times \frac{r \times n}{|r \times n|}}{|n|} \tag{8.45}$$

The total pressure, based on the density at the center of the panel, is

$$F = \frac{TA}{R^2 \Omega_{\text{eff}}} (\cos^4 \theta + 0.06) \cos^4 \theta \tag{8.46}$$

where T is the thruster thrust, R is the distance from the thruster to the center-of-pressure, Ω_{eff} is the effective plume angle in radians (about $\pi/2$), and θ is as defined in the figure. The normal force is

$$F_N = P((2-\sigma_N)\sin\phi + \sigma_N\beta)\sin\phi \tag{8.47}$$

and the tangential force is

$$F_T = P\sigma_T \cos\phi \sin\phi \tag{8.48}$$

where

$$\beta = \sqrt{\frac{\pi T_P}{T_0}\left(\frac{\gamma-1}{4\gamma}\right)} \tag{8.49}$$

where σ_T and σ_N are the surface accommodation coefficients, T_P is the surface temperature of the panel, γ is the ratio of specific heats for the plume gas, and T_0 is the combustion temperature of the plume. The accommodation coefficients are generally a function of ϕ but can be set equal to 1 in the model to get the largest plume pressure.

8.3.12 Outgassing force

Outgassing is due to material loss due to diffusion from the spacecraft. The diffusion equation is [7]

$$\frac{dm}{dt} = -\frac{m}{\tau} \tag{8.50}$$

where

$$\tau = \tau_0 e^{\frac{E}{RT}} \tag{8.51}$$

where R is the universal gas constant for that molecule and T is temperature. The activation energy E is a function of the material but the reaction time constant τ_0 must be determined experimentally. Outgassed material may also impact a spacecraft surface and will stick, at least temporarily, to that surface. Outgassing has been responsible for spacecraft perturbations, including that on the Microwave Anisotropy Satellite [8].

The thrust due to outgassing is

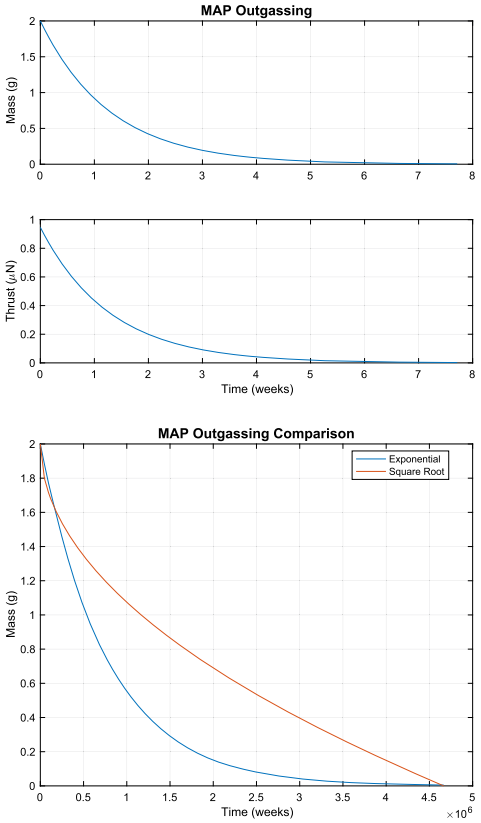
$$T = \dot{m}u_e \tag{8.52}$$

where u_e is the velocity of the material. The exhaust velocity is [8]

$$u_e = \sqrt{\frac{2k_b T}{N\mathfrak{M}}} \tag{8.53}$$

where $\mathfrak{M}$ is the molecular mass, $k_b = 1.38 \times 10^{-23}$ J/K, T is the temperature, and N is the atomic number of the outgassed material. Example 8.9 gives an example from the Microwave Anisotropy Spacecraft for water outgassing.

```
kB       = 1.38e-23;
T        = 150;
m        = 1.7e-27;
N        = 18;
uE       = sqrt(2*kB*T/(m*N));
m0       = 0.002;
tau      = 9*86400;
t        = linspace(0,6*tau);
m        = m0*exp(-t/tau);
mDot     = m/tau;
thrust   = mDot*uE*1e6;
R        = 461.52;
EA       = 26.4e3; % J/mole
tau0     = tau/exp(R*T/EA);
[tS,tL]  = TimeLabl(t);
yL       = {'Mass (g)','Thrust (\muN)'};
Plot2D(tS,[m*1000;thrust],tL,yL,'MAP Outgassing')

g0       = 0.5*m0/(exp(EA/(R*T))*sqrt(6*tau));
mS       = m0 - 2*g0*exp(EA/(R*T))*sqrt(t);
Plot2D(t,[m;mS]*1000,tL,yL{1},'MAP Outgassing
       Comparison','lin', ...
[],[],[],[],[],{{'Exponential' 'Square Root'}})
```



Example 8.9: Outgassing from the Microwave Anisotropy Spacecraft. Both exponential and square-root models are shown.

τ_0 is 56 483 s. Soares [7] notes that diffusion has been experimentally shown to have a $\sqrt{t}$ time dependence. The outgassing mass is

$$m = m_0 - 2q_0 e^{\frac{E}{RT}} \sqrt{t} \tag{8.54}$$

This, however, leads to an infinite diffusion rate at $t = 0$ so is not useful for disturbance analyses.

8.3.13 Shadowing

An issue with external disturbances produced by external fluxes, whether aerodynamic or radiation, is that all surfaces will not be exposed to the flux because they are shadowed

by other surfaces. The problem of shadowing is encountered in computer graphics. It is solved by z-buffering, also known as depth buffering, which is simply drawing the objects from back to front along the view axis of the camera. For disturbance analyses, we need to be able to determine what part of a surface is visible behind a blocking surface. A simple, and reasonably fast algorithm, is the scan-line algorithm. This is also known as ray tracing. In this algorithm, the object is projected onto a plane normal to the flux vector and scanned. The objects are sorted, like in z-buffering, from front to back. If the scan line intercepts a front surface that surface is included. The scans that do not intercept the front surface are used to determine intersections with the next surface, and so forth. Fig. 8.11 shows an example with two triangles.

This method can be computationally intensive. The scan-line pattern must be chosen so that a scan line will pass through each triangle of a surface at least once. Complex geometries may cause scanning artifacts. The need for the computation of shadowing must be determined on a case-by-case basis. For really simple geometries, such as a CubeSat with a single deployable solar wing, it may be easier to analytically compute the shadowing.

8.4. Internal disturbances

Table 8.3 lists major internal disturbances on spacecraft. Internal disturbances can cause jitter, which is high-frequency torques internal to the spacecraft.

The modeling of internal disturbances depends on the frequency of the disturbance. For example, with a reaction wheel, cogging torques, or imbalance torques are a function of the angular speed of the wheel. This varies as the wheel spins up or down. At low speeds, the torque might be within the bandwidth of the control system and would be modeled as a harmonic disturbance. At higher speeds, the torque would be modeled as noise. Internal dynamical disturbances can be modeled with multibody dynamics. For example, slosh motion during a burn can be modeled as a single or double pendulum. Friction torques for the reaction wheel are computed as part of the reaction-wheel model.

Thermal snap is due to the rapid change of temperature on a deployed part of the spacecraft, such as a solar wing or antenna causing it to vibrate. The Upper Atmosphere Research Satellite experienced thermal snap due to its single solar wing [9]. This effect can be modeled with multibody dynamics or as a single wheel with a harmonic torque.

8.5. Fourier-series representation

Fourier series can be used to represent the disturbances on a spacecraft. A complex numerical disturbance model is run and the output Fourier is analyzed to generate a harmonic series. The series can be used for simulations or actuator sizing. The Fourier

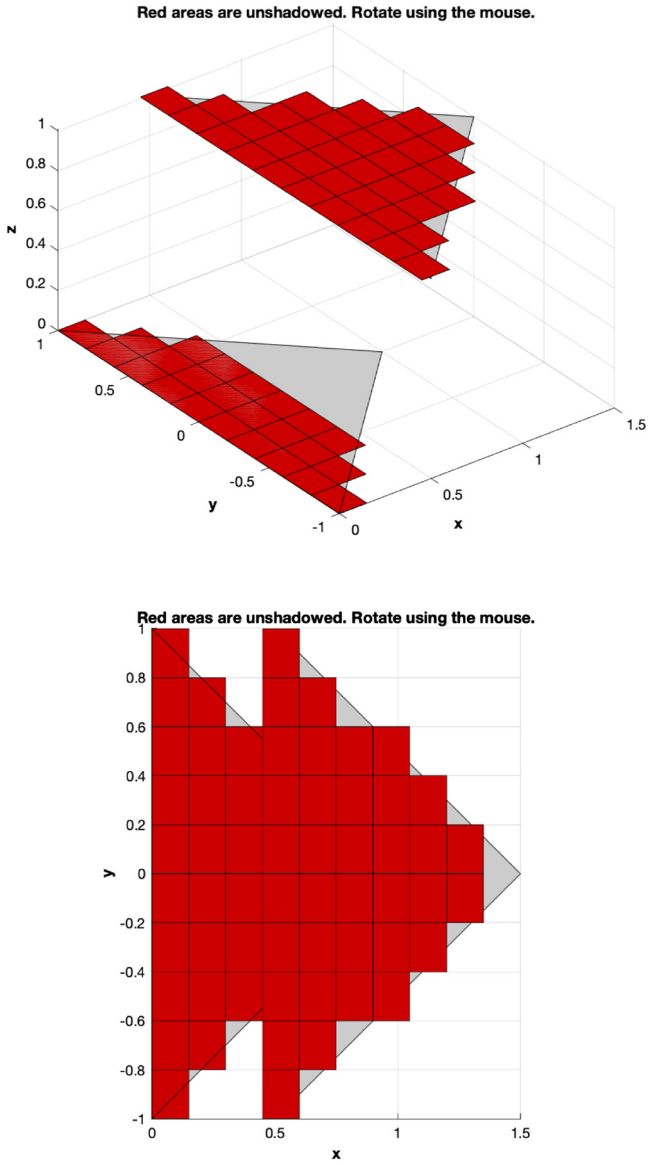


Figure 8.11 Scan-line algorithm. The red rectangles are the included areas. The picture on the right shows the view down the flux vector.

series also gives insight into the nature of the disturbances.

$$T = a_o + \sum_k a_k \cos(knt) + b_k \sin(knt) \tag{8.55}$$

Table 8.3 Internal disturbances.

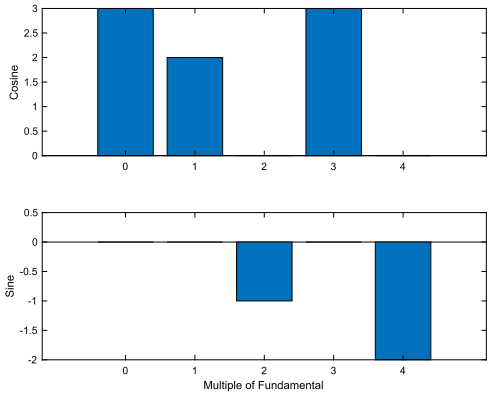
Type	Source	Characteristics
Dynamic imbalance	Rotating component with off-diagonal inertia terms.	Periodic with the rotation rate.
Friction	Rotating or sliding parts.	Has three major components: stiction, or starting friction; Coulomb friction, which is independent of rate; and viscous, which is proportional to the rate. Turbulent damping is proportional to the rate squared.
Motor torques and forces	Motors.	Articulating components.
Moving parts	Dynamics of internal moving parts.	Torques are proportional to the square of relative rates.
Rotors	Momentum of rotors.	Gyroscopic torques.
Rotor imbalances	Center-of-mass offset of a rotating component.	Periodic with the rotation rate.
Slosh	Liquids.	Movement of liquids within the spacecraft. Important for liquid-fueled upper stages.
Thermal snap	Sudden structural deformation.	Caused by rapid heating or cooling.

where n is the orbit period for a planet-pointing spacecraft. Any period associated with the spacecraft can be used. This is convenient for long-duration studies, particularly of momentum-control systems. It assumes that the spacecraft attitude motion is not coupled with the disturbances, which is true when a spacecraft is pointing tightly at a target.

The discrete Fourier transform is a very efficient algorithm for generating the Fourier transform. Fast Fourier Transform (FFT) code is available in virtually every programming language. Example 8.10 shows how it is used. The code generates a vector of length n^2. The vectors are computed with sine and cosine terms with frequencies that are multiples of the fundamental frequency. The FFT is computed and half the values are discarded because they are duplicates of the first half. Each value corresponds to a harmonic that is a multiple 4 of the fundamental. The bias and cosine terms are from the real part of f. The sine terms are the negatives of the imaginary part. The coefficients are recovered exactly, even in the presence of noise. f is zero if there are no harmonics.

This can be applied to the problem of a geosynchronous satellite with a Sun-pointing solar wing and an Earth-pointing bus. This is shown in Example 8.11. The model has only five surfaces. Eclipses are included. A Fourier series is generated for the force. The spacecraft is modeled as a solar wing that is inertially fixed and a bus that is fixed with

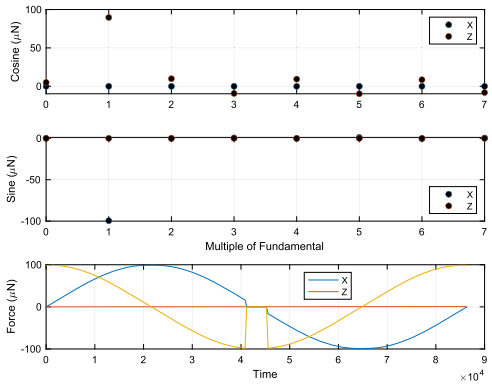
```
%% Fourier Series

period  = 10;
omega   = 2*pi/period;
n       = 2^12;
t       = linspace(0,period,n+1);
y       = 3 + 2*cos(  omega*t)...
            -   sin(2*omega*t)...
            + 3*cos(3*omega*t)...
            - 2*sin(4*omega*t) + randn(1,n+1);
f       = fft(y,n);
f       = f(1:n/2);
k       = 0:(n/2-1);
a       = 2*real(f)/n;
a(1)    = a(1)/2;
b       = -2*imag(f)/n;

figure('name','Fourier Coefficients')
subplot(2,1,1)
bar(k(1:5),a(1:5))
ylabel('Cosine')

subplot(2,1,2)
bar(k(1:5),b(1:5))
xlabel('Multiple of Fundamental')
ylabel('Sine')
```



Example 8.10: Fourier coefficients of a harmonic function.

respect to LVLH. Spring equinox is chosen as the time of year so that the Sun is in the LVLH xz-plane. The spacecraft is in geosynchronous orbit and eclipses are modeled.

```
period          = 86400; % s
areaWing        = 2*10; % m^2
areaBus         = 2*2; % m^2
rhoW            = [0.5 0.1 0.3];
rhoB            = [0.5 0.4 0.1];
u               = [1 -1 0 0;0 0 0 0; 0 0 1 -1];
p               = 1367/3e8;
n               = 2^8;
t               = linspace(0,period,n+1);
jD              = 2464042.0515 + t/86400;
[uSun,rSun]     = SunV1(jD);
[r,v]           = RVOrbGen([42167 0 0 0 0 0],t);
e               = Eclipse( r, uSun.*rSun );
q               = QLVLH(r,v);
s               = uSun.*e;
%% Compute the force in the ECI frame
fW              = -p*areaWing*(rhoW(3) + (4/3)*rhoW
        (2) + rhoW(1))*s;
sB              = QForm(q,s);
fB              = zeros(3,n+1);
%% Cycle through the faces
for k = 1:4
   uK     = u(:,k);
   uS     = uK'*sB;
   uS(uS < 0) = 0;
   fB     = fB - p*areaBus*uS.*((rhoW(3)*uS + rhoW
            (2)/3).*uK + (rhoW(2)+rhoW(3))*sB);
end
%% Fourier analysis
fT      = (fB + QForm(q,fW))*1e6;
f       = fft(fT',n)';
f       = f([1 3],1:n/2);
k       = 0:(n/2-1);
a       = 2*real(f)/n;
a(:,1)  = a(:,1)/2;
b       = -2*imag(f)/n;
```



Example 8.11: Fourier coefficients for a spacecraft in geosynchronous orbit.

The Fourier code is the same as the code in the example above. The force is dominated by the wing, which produces a sinusoidal force, even with the discontinuity due to the eclipse. The eclipse is evident in the middle of the plot when the Sun is on the opposite side of the Earth from the spacecraft. The force is dominated by the first harmonic, which is the orbit-rate term. The *Y* forces, which are zero because the Sun is in the *xz*-plane, are not plotted.

References

[1] R.M. Fredo, A Numerical Procedure for Calculating the Aerodynamic Coefficients for Complex Spacecraft Configurations in Free-Molecular Flow, Ph.D. thesis, The Pennsylvania State University, 1980.
[2] J.A. Storch, Aerodynamic Disturbances on Spacecraft in Free Molecular Flow, Tech. Rep., Space and Missiles Systems Division, October 2002.
[3] G. Mead, D. Fairfield, A quantitative magnetosphere model derived from spacecraft magnetometer data, Journal of Geophysical Research 80 (4) (February 1975) 523–534.
[4] Andrew Marshall, S.B. Luthcke, Radiative force model performance for TOPEX/Poseidon precision orbit determination, The Journal of the Astronautical Sciences 42 (2) (April-June 1994) 229–246.
[5] P.C. Knocke, J.C. Ries, B.D. Tapley, Earth radiation pressure effects on satellites, in: AIAA/AAS Astrodynamics Conference, AIAA, August 1988, pp. 577–583, AIAA-88-4292.
[6] I. Ashdown, Radiosity: A Programmer's Perspective, John Wiley and Sons, 1994.
[7] A.Y. Huang, G.N. Kastanas, L. Kramer, C.E. Soares, R.R. Mikatarian, Materials outgassing rate decay in vacuum at isothermal conditions, in: J. Egges, C.E. Soares, E.M. Wooldridge (Eds.), Systems Contamination: Prediction, Control, and Performance 2016, in: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 9952, Sept. 2016, p. 995206.
[8] S. Starin, J. O'Donnell, D. Ward, E. Wollack, P. Bay, D. Fin, An anomalous force on the MAP spacecraft, in: AIAA Guidance Navigation and Control Conference, Aug. 2002.
[9] M. Lambertson, D. Rohrbaugh, J. Garrick, Solar array thermal snap and the characteristics of its effect on UARS, in: Flight Mechanics/Estimation Theory Symposium, Feb. 1993, pp. 575–588.

CHAPTER 9

Budgets

9.1. Introduction

This chapter describes pointing, propellant, and mass budgets. Budgets are a way of keeping track of the factors that contribute to each characteristic of a spacecraft.

Pointing budgets quantify the effect of each error source on the overall vehicle pointing. Propellant budgets quantify the contribution of each maneuver involving thrusters to the consumption of fuel. On most spacecraft, propellant budgets are dominated by the fuel consumption to perform orbital maneuvers. Mass budgets provide information on the total mass and inertia matrix of a satellite. Power budgets determine overall spacecraft power consumption and are needed to size the power sources, usually solar arrays, and batteries.

9.2. Pointing budgets

There are several competing methodologies for doing pointing budgets for satellites. One of the most popular is to divide the error contributions into four temporal categories:

1. Bias (e.g., manufacturing misalignment that is fixed throughout the mission);
2. Long-term (e.g., seasonal variation in the yaw body fixed torque on a momentum-bias spacecraft without yaw sensing);
3. Diurnal (e.g., thermally induced rotation);
4. Short term (e.g., random noise);

and treat all contributions within each temporal category as if they were normally distributed uncorrelated random variables. The contributions within each category are then summed (by taking the square root of the sum of their squares). The totals for each temporary category are then added to get the final per-axis numbers.

Other methodologies examine each contribution and determine its nature, separating deterministic and other contributions from random. For example, orbit errors should be separated from alignment errors. This provides somewhat more conservative but statistically more justifiable results.

Take for example a geosynchronous communications satellite. This type of satellite points at the Earth for its entire mission life. The yaw axis is aligned with nadir, the pitch axis points South, and the roll axis points East. If the spacecraft has a single circular beam about nadir only roll and pitch errors matter. If it has off-nadir spot beams or if the main beam has a nonzero elevation, yaw couples into the beam errors. Once roll,

https://doi.org/10.1016/B978-0-32-399915-1.00020-6

pitch, and yaw error values have been computed, they must be combined to form the beam-pointing accuracy. The beam is the radio-frequency wavefront that touches the planet. The beam-pointing accuracy is

$$\phi_{AZ} = \theta_{roll} + \frac{\delta_{AZ}}{180/\pi}\theta_{yaw} \tag{9.1}$$

$$\phi_{EL} = \theta_{pitch} + \frac{\delta_{EL}}{180/\pi}\theta_{yaw} \tag{9.2}$$

where δ_{EL} and δ_{AZ} are the beam-center offsets.

An example pointing budget is shown in Table 9.1. This is for the control mode using magnetic torquers and an Earth sensor. SA is the solar array. MWA is the

Table 9.1 Normal pointing mode pointing budget.

#	Item	Roll	Pitch	Yaw	Units
Bias					
1	ESA temperature effects	0.0150	0.0150	0.0000	deg
2	ESA optics w.r.t. cube	0.0250	0.0250	0.0000	deg
3	ESA cube to MRC	0.0250	0.0250	0.0000	deg
4	Antenna B.S. w.r.t. antenna RC	0.0180	0.0180	0.0000	deg
5	Antenna RC w.r.t. MRC	0.0150	0.01500	0.0000	deg
6	MWA spin axis w.r.t. mirror	0.0060	0.0060	0.0000	deg
7	MWA mirror w.r.t. MRC	0.0170	0.0170	0.0000	deg
8	Attitude sensing (Earth radiance)	0.0012	0.0012	0.0000	deg
9	S/C disturbance torque	0.0031	0.0076	0.0000	deg
	Subtotal	0.0486	0.0491	0.0000	deg
Diurnal					
10	East/West position	0.0100	0.0100	0.0000	deg
11	Attitude sensing (Earth radiance)	0.0100	0.0100	0.0000	deg
12	Orbit inclination	0.0100	0.0000	0.0100	deg
13	S/C disturbance torque	0.0080	0.0000	0.1470	deg
14	Antenna thermal distortion w.r.t. ESA RC	0.0100	0.0100	0.0000	deg
15	Thermal distortion MWA w.r.t. ESA RC	0.0200	0.0500	0.0000	deg
	Subtotal	0.0294	0.0529	0.1473	deg
Short Term					
16	Attitude sensor & actuator noise/resolution	0.0200	0.0200	0.0000	deg
17	SA stepping transients	0.0000	0.0500	0.0000	deg
18	S/C disturbance torques	0.0000	0.0000	0.0000	deg
	Subtotal	0.0200	0.0539	0.0000	deg
	Total	0.0980	0.1558	0.1473	deg
CEP = 0.1707 deg					

momentum-wheel assembly. ESA is the Earth Sensor Assembly. S/C is spacecraft. MRC is the master reference cube that is the master alignment reference for the spacecraft. This is a cube that reflects laser light from a laser measuring device. CEP is the Circular Error Probability. The CEP in degrees says that for normally distributed random errors the 3-sigma probability is that the error will be within the CEP. This budget was generated for a momentum-bias spacecraft that uses an Earth sensor (ESA) to measure roll and pitch. The quantity of interest is the beam alignment (antenna boresight or B.S.) with respect to a target on the Earth. The temporal quantities are: bias, assumed fixed for the life of the spacecraft; diurnal, which vary with the orbit period; and short term, which changes at rates faster than orbit rate. Yaw is not sensed and is controlled solely by the spacecraft momentum. The largest component, in this case, is the orbit-rate disturbance so the only entries for yaw are orbit inclination and this disturbance-related error. In roll and pitch, there are many errors due to misalignments. Alignments are measured using an optical system. ESA RC is the reference cube for the Earth sensor.

ESA errors appear in all three temporal categories. These are taken from the manufacturer's specifications. All alignment errors are determined by measuring postvibration errors. The spacecraft is aligned and then placed on a shaker to simulate launch. The change in alignments is considered the alignment uncertainties.

9.3. Propellant budgets

Propellant budgets are detailed accounts of the propellant needed for all spacecraft operations. They include fuel needed for all orbit-change operations and for attitude control operations that may include pointing control, momentum unloading, reorientations, etc.

Attitude control fuel consumption can be divided into components that are proportional to the orbit-change fuel consumption and those that are not. The former is due to misalignments in the delta-V engines or asymmetries of multiple delta-V engines with respect to the center-of-mass. The attitude control system must cancel the disturbance torques due to the delta-V engines. The attitude control fuel requirement is usually found through simulation. A simulation can provide the torque and force demands for typical maneuvers. These are then used to compute the required pulsewidth demand and then the impulse that then leads to the fuel used per maneuver. For a blowdown propellant system, the thrust drops with time and this must be considered.

The second component includes firings that are not proportional to the magnitude of the delta-V. Both components must be determined through analysis and simulation. Orbit operations that involve changing the velocity of the spacecraft include:

1. Station acquisition;
2. Station keeping;
3. Station changes;

4. Deorbit.

Station keeping includes inplane and out-of-plane maneuvers. Station changes are generally within the plane of the orbit. Deorbit maneuvers include forcing an early reentry of the satellite for low-Earth orbit satellites and maneuvers to push geosynchronous satellites out of a geosynchronous orbit.

Table 9.2 shows an example propellant budget. This shows a budget for a hypothetical geosynchronous spacecraft through its second year of life. There are three thruster systems used. The mission-orbit system is monopropellant hydrazine and is divided into two half-systems that are pressurized independently. The third system is the solid rocket motor.

9.4. Power budgets

The attitude control system consumes power. The power consumption can be divided into standby power and active power. Standby power is the power consumed by a device when it is on but not active. For example, when a reaction wheel is enabled the power electronics will consume power. When the wheel is used for control the generation of torque will consume power. For a hydrazine monopropellant thruster, the catalyst-bed heaters consume power even if the thruster is not firing.

The active power consumed by the flight processors will be proportional to the processing load. You experience this with a laptop that can become uncomfortably hot when a computationally intense process is underway. Flight processors are shared so the attitude control power budget should only include the ACS contributions.

Different modes of operation will have different power needs. A power budget is needed for each mode. Each device should have a line listing standby, average, and maximum power consumption.

Maneuvers 1 through 4 prepare for the Apogee Kick Motor (AKM) firing. This rocket engine is used to put the spacecraft into geosynchronous orbit. They include a spin-precession maneuver (SPM) from negative orbit normal (NON) to the AKM burn attitude, a spin-up to the AKM spin rate (needed to stabilize the vehicle), and a small SPM trim. All use of the solid occurs during the AKM burn. This is followed by more attitude maneuvers to prepare for acquisition. Maneuvers 10 and 11 are drift-orbit maneuvers designed to get the spacecraft to the station. Usually, more than two are required. The remaining maneuvers are to correct East/West drift and inclination drift.

The inclination maneuvers use electrothermal hydrazine thrusters (EHT) so the specific impulses (the measure of fuel consumption per second) for delta-V are much higher. In all cases, the budget accounts for the different specific impulses for the attitude control thrusters and the delta-V thrusters.

Table 9.2 Example propellant budget.

Item	Description	Total	ACS Isp	DV Isp	Fuel Remaining	Fuel Remaining	Fuel Remaining	Fuel Used	Fuel Used	Fuel Used
1	Initial State	2055.92	0.00	0.00	117.89	117.89	865.00	0.000	0.000	0.000
2	SPM NON to AKM	2055.82	100.50	0.00	117.84	117.84	865.00	0.051	0.051	0.000
3	Spin Up 5 to 60 rpm	2051.14	118.33	0.00	115.50	115.50	865.00	2.341	2.341	0.000
4	SPM Trim	2051.13	100.10	0.00	115.50	115.50	865.00	0.001	0.001	0.000
5	AKM Firing	1188.56	0.00	265.00	115.50	115.50	2.43	0.000	0.000	862.574
6	Despin to 5 rpm	1183.88	118.33	0.00	113.02	113.30	2.43	2.481	2.202	0.000
7	SPM AKM to PON	1183.43	102.51	0.00	112.79	113.07	2.43	0.225	0.226	0.000
8	Despin to 2.53 rpm	1183.22	118.33	0.00	112.68	112.97	2.43	0.111	0.099	0.000
9	Despin from 1.1 rpm to orbit rate	1183.15	118.33	0.00	112.64	112.94	2.43	0.037	0.033	0.000
10	Drift-orbit station acquisition	1174.15	100.00	210.00	108.14	108.44	2.43	4.500	4.500	0.000
11	Drift-orbit inclination correction	1163.94	100.02	360.00	103.05	103.32	2.43	5.093	5.115	0.000
12	East/West 1997	1163.89	100.00	143.01	103.03	103.30	2.43	0.022	0.022	0.000
13	East/West 1998	1163.85	100.00	143.02	103.00	103.28	2.43	0.022	0.022	0.000
14	North/South 1998	1159.97	100.02	360.00	101.07	101.33	2.43	1.936	1.943	0.000
15	East/West 1998	1159.92	100.00	144.29	101.05	101.31	2.43	0.022	0.022	0.000
16	East/West 1998	1159.88	100.00	144.30	101.02	101.29	2.43	0.022	0.022	0.000
17	East/West 1998	1159.84	100.00	144.31	101.00	101.27	2.43	0.022	0.022	0.000
18	North/South 1998	1155.97	100.02	360.00	99.07	99.33	2.43	1.930	1.936	0.000
19	East/West 1998	1155.93	100.00	145.57	99.05	99.31	2.43	0.022	0.022	0.000
20	East/West 1998	1155.88	100.00	145.58	99.03	99.29	2.43	0.022	0.022	0.000
21	East/West 1998	1155.84	100.00	145.60	99.01	99.27	2.43	0.022	0.022	0.000

9.5. Mass budgets

A mass budget includes the masses of all components of the vehicle and the inertias of all of those components. The budget includes the total of the masses and inertias. While masses are relatively easy to determine, inertias are not. Inertias can be calculated by using the mass distributions of components and the locations of those components

$$I = -\sum_{k-1}^{N} m_k (r_k - c)^{\times} (r_k - c)^{\times} \tag{9.3}$$

where r is the position vector of the component with respect to a reference, m_k is the mass of the component, and c is the position of the center-of-mass where

$$0 = \sum_{k-1}^{N} m_k (r_k - c) \tag{9.4}$$

This assumes that each mass can be approximated as a point mass. A better approximation is to enclose the component in a rectangular envelope and compute the inertia from the formula

$$I = \frac{m}{12} \begin{bmatrix} b^2 + c^2, 0, 0 \\ 0, a^2 + c^2, 0 \\ 0, 0a^2 + b^2 \end{bmatrix} \tag{9.5}$$

where a is the x-length, b is the y-length, and c is the z-length. m is the total mass of the box. For larger components the inertia matrix can be computed on a test rig by applying 3-axis torques and measuring the resulting acceleration. Most CAD packages can also compute inertias.

When combining inertias to compute the total vehicle inertia it is necessary to know the alignments. Since alignment measurements have uncertainty, the best that will be known is the inertia to within the alignment errors.

One difficulty with creating a total spacecraft inertia matrix is that suppliers rarely provide an inertia matrix for a component. If the component is small its inertia matrix contribution is

$$I_k = -m_k d_k^{\times} d_k^{\times} \tag{9.6}$$

If a component is larger, this can be improved by computing the inertia matrix of a box that encloses the component. The mass of the component is then multiplied by the 3 × 3 inertia matrix for a uniform box.

CHAPTER 10

Actuators

10.1. Space story

The new Series 5000 satellites had a PID controller for station keeping. On the first station-keeping maneuver, we noticed that it was oscillating. The problem was that there was a high-gain instability, which can happen with a PID because, at the minimum pulsewidth, where the valve is opened for the shortest possible time, the thruster produced more impulse than expected. This was fixed by adjusting the PID gains. This was the first time we had noticed this problem.

10.2. Introduction

The performance of a spacecraft-control system is limited by the performance of its sensors and actuators. This chapter discusses the types of actuators that are used in spacecraft-control systems. Many of the actuators are used just for attitude control. Others are used for orbit adjustment and station keeping, as well as attitude control.

10.3. Types of actuators

Table 10.1 lists some actuators used in spacecraft. We refer to attitude control and momentum unloading. Systems that rely on momentum exchange (reaction wheels, momentum wheels, and control-moment gyros) need momentum unloading, that is an inertial torque, to compensate for inertially fixed-attitude disturbances.

Table 10.1 Classes of Actuators.

Type	Description	Application
Magnetic torquer air coil	Wire wrapped around the frame of the spacecraft to form a planar loop.	Momentum unloading, two-axis control. Average three-axis control.
Magnetic-torquer bar	Wire wrapped around a magnetic steel core, often multiple coils around a single core. Iron core torquers are usually operated in saturation. The second coil can produce 20% more torque, which is useful in an emergency.	Momentum unloading, two-axis control. Average three-axis control.

continued on next page

https://doi.org/10.1016/B978-0-32-399915-1.00021-8

Table 10.1 (*continued*)

Type	Description	Application
Thruster hydrazine	Monopropellant hydrazine is passed through a catalyst bed that causes an exothermic reaction. This produces thrust.	Attitude and orbit control.
Thruster bipropellant	An oxidizer and fuel are mixed to produce a chemical reaction.	Attitude and orbit control.
Thruster cold gas	Pressurized gas is expanded through a nozzle.	Attitude Control.
Electrothermal hydrazine thrusters (EHT)	Same as hydrazine except that it has a heating coil to heat the exhaust of the exothermic reaction.	Attitude control.
Thruster arc jet	Same as hydrazine except that it has an electric arc to heat the exhaust of the exothermic reaction.	Orbit changes.
Pulsed plasma thruster (PPT)	Ionized gas is accelerated in a magnetic field.	Attitude and orbit control.
Hall thruster	Thruster uses the Hall effect to accelerate ions.	Orbit control.
Ion engine	Thruster in which ions are accelerated using a charged grid.	Orbit control.
Magnetoplasmadynamic Thruster (MPD)	Magnetic field accelerates ions. Lithium is the only practical fuel.	Orbit control.
Variable Specific Impulse Magnetoplasma Rocket (VASIMR)	An electric rocket composed of three cells. The first creates a plasma. The second further heats it with radio-frequency waves the third is a magnetic nozzle that converts it to a directional flow separate from the magnetic-field lines.	Orbit control.
Reaction wheel	A wheel is attached to the spacecraft via an electric motor. Provides a means of regulated momentum exchange between a flywheel and the rest of the spacecraft through reaction torques. The motor uses current feedback to linearize the torque response. This compensates for back-electromotive force.	Attitude control.
Momentum wheel	A wheel is attached to the spacecraft via an electric motor. Provides a means of regulated momentum exchange between a flywheel and the rest of the spacecraft through reaction torques. The wheel spins at a high rate to provide a momentum bias. It is modulated around this speed to control the attitude along the spin axis.	Attitude control.

continued on next page

Table 10.1 (*continued*)

Type	Description	Application
Pivoted wheel	Reaction wheel on a single or two-degree-of-freedom platform A pivoted wheel has a limited range of angular motion, in contrast with a control-moment gyro that has large-angle rotation capability. In a pivoted wheel both the rotation of the wheel about its axis and the rotation of the wheel assembly is used for control.	Attitude control.
Control-moment gyro	A spinning wheel is attached to the spacecraft through one or two single-degree-of-freedom hinges. The spin rate is maintained and torques are generated normally only about the pivot axes. Some systems also vary the wheel speed for an additional control degree-of-freedom.	Attitude control of slew maneuvers.
Solar paddles	Solar-pressure torques are generated by moving the paddles. This is normally done with solar panels or with solar sails.	Attitude control and momentum unloading.

10.4. Reaction-wheel model

10.4.1 Introduction

Reaction wheels are momentum-exchange devices. A motor is fixed to the spacecraft and the shaft of the motor is attached to a flywheel. When a control voltage is applied to the motor, the motor generates a torque spinning the wheel in one direction and the spacecraft in the other; hence the term reaction wheel. Since the torque is internal to the spacecraft system, the reaction wheel cannot change the total inertial angular momentum of the spacecraft system. Rather, it transfers momentum between the flywheel and the rest of the spacecraft. If the external torque on the spacecraft is periodic (with a sufficiently high frequency) with respect to the inertial frame, then the reaction wheel can completely control the spacecraft. However, if there is a constant inertial torque, the wheel will spin up and eventually saturate. The wheel can also be used to maneuver the spacecraft. When the wheel spins up the spacecraft will develop an angular rate. When the wheel spins down it will absorb the angular momentum in the body and stop the motion. See Fig. 10.1.

Fig. 10.2 shows the anatomy of an axial-flux reaction wheel. It uses a three-phase stator and a Halbach array rotor. The angle of the rotor is measured from the currents in the stator and the measured angle from the angle encoder. Axial flux means that the magnet flux is along the spin axis. The Halbach magnet configuration concentrates the flux on one side of the magnet, increasing the available torque.

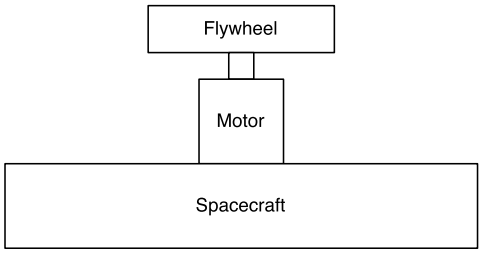


Figure 10.1 Reaction wheel.

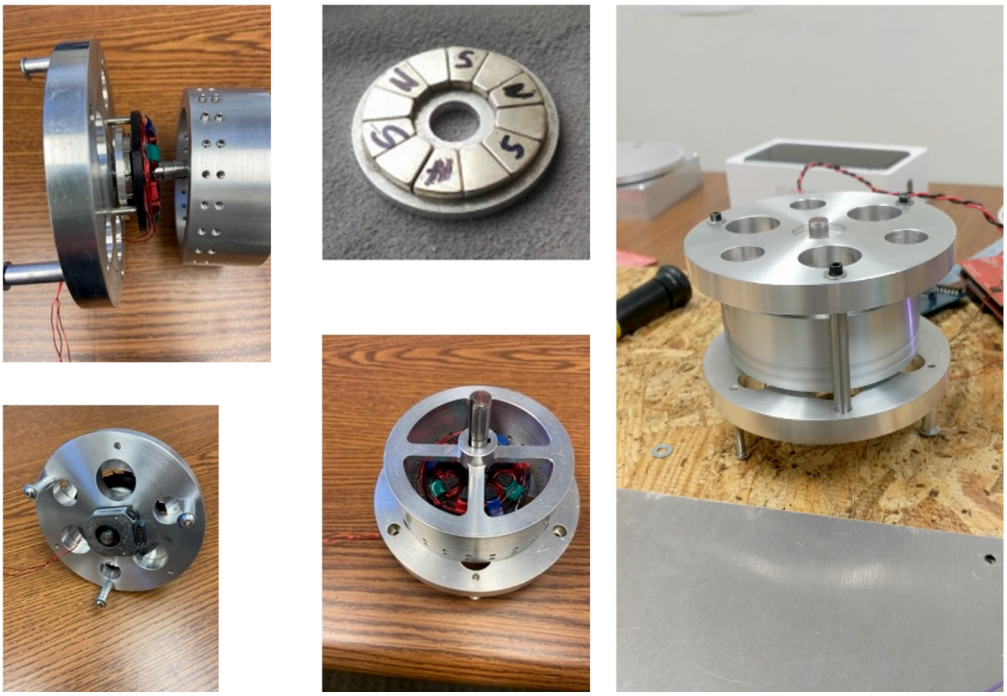


Figure 10.2 Anatomy of an axial-flux motor reaction wheel. Clockwise from left: side view showing the stator coils, the Halbach rotor magnet array, the complete wheel in operation, view from the top and bottom view showing the optical angle encoder.

10.4.2 Momentum exchange

The single-axis reaction wheel and spacecraft model is

$$H = I\omega + J(\Omega + \omega) \tag{10.1}$$

$$T = \dot{H} = I\dot{\omega} + J(\dot{\Omega} + \dot{\omega}) \tag{10.2}$$

$$T_w = J(\dot{\Omega} + \dot{\omega}) \tag{10.3}$$

where H is the inertial angular momentum, T is the body torque, I is the body inertia, J is the wheel inertia, Ω is the wheel angular rate, and ω is the body angular rate.

If the attitude control system is perfect then

$$\dot{\omega} = 0 \tag{10.4}$$

$$T = J\dot{\Omega} \tag{10.5}$$

The wheel absorbs all of the momentum from the external torque.

10.4.3 Motor model

Most reaction wheels are driven by DC motors. A simplified model is shown in Fig. 10.3. The incoming torque command is converted into a current by the first block.

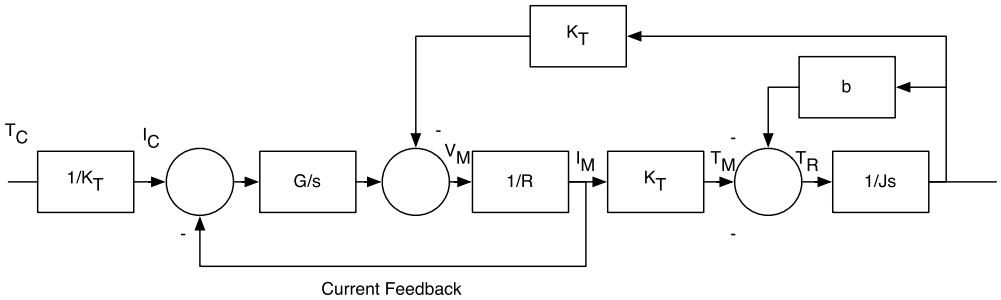


Figure 10.3 DC motor model.

The summing junction subtracts the motor current. This current feedback loop compensates for the back-electromotive force that is a function of speed. This permits the device to be used as a torque actuator.

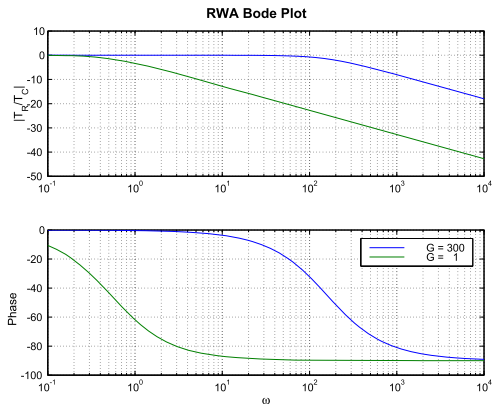
The difference between the commanded current and feedback current is integrated and multiplied by a gain. The back-electromotive force (back emf) voltage is subtracted, resulting in the motor voltage. The motor voltage is limited by the spacecraft bus voltage. The motor voltage is divided by the motor resistance to get the motor current, which is then multiplied by the motor gain to produce the motor torque. The frictional torques are subtracted from this to get the reaction torque. When using the wheel as a torque actuator, the transfer function from commanded torque to reaction torque is of interest. The current loop compensates for the back-emf. To compensate for the friction torques, another loop would be needed that fed back rate. This is often done in robotics where precise torque control is necessary.

The transfer function from commanded torque to reaction torque is

$$\frac{T_R}{T_C} = \frac{G}{R} \frac{s}{s^2 + \left(\frac{b}{J} + \frac{k_T^2}{RJ} + \frac{G}{R}\right)s + \frac{bG}{JR}} \tag{10.6}$$

where J is the reaction-wheel inertia, b is the viscous damping coefficient, R is the motor resistance, k_T is the current to motor torque gain, and G is the forward gain. There is no DC response since when a constant command is supplied the wheel spins up and the viscous friction torque eventually cancels the motor torque. At high frequencies, the RWA gain rolls off as $1/s$. At intermediate frequencies, if G is large, the gain is nearly 1.

```
i         = 0.0398; kT = 0.0329; r = 1.8856; b =
        0.001;
g         = 300;
num       = [g/r 0];
den       = [1 b/1i+kT^2/(r*1i) + g/r b*g/(r*1i)];
w         = logspace(-1,4);
[m1, p1] = FResp( num, den, w );
g         = 1;
num       = [g/r 0];
den       = [1 b/1i+kT^2/(r*1i) + g/r b*g/(r*1i)];
[m2, p2] = FResp( num, den, w );
m1        = 10*log10(m1); m2 = 10*log10(m2);
h         = Plot2D( w, [m1;m2;p1;p2], '\omega',...
              {'|T_R/T_C|' 'Phase'},'RWA Bode Plot'
                 ,...
              'xlog',{'[1 2]' '[3 4]'});
legend('G = 300','G = 1');
```



Example 10.1: Reaction-wheel frequency response.

10.4.4 Reaction-wheel state equations with current feedback

There are two states in this model, the wheel speed (a mechanical state) and the integral of the current difference (an electrical state). The state equations are derived as follows. The reaction torque is

$$T_R = k_T i_M - b\omega - T_F \tag{10.7}$$

where T_F is all nonviscous friction torques. The motor current is

$$i_M = \frac{1}{R}(V_M - k_T\omega) \tag{10.8}$$

The motor torque is

$$V_M = G\int (i_M - i_C) \tag{10.9}$$

The electrical state derivative is

$$\dot{z} = i_M - i_C \tag{10.10}$$

The mechanical state derivative is

$$\dot{\omega} = \frac{T_R}{J} \tag{10.11}$$

The state equations are

$$\begin{bmatrix} \dot{\omega} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \frac{k_T^2}{RJ} & \frac{bk_T G}{J^2 R} \\ \frac{k_T}{R} & \frac{G}{R} \end{bmatrix} \begin{bmatrix} \omega \\ z \end{bmatrix} + \begin{bmatrix} 0 & 1/J \\ 1/k_T & 0 \end{bmatrix} \begin{bmatrix} T_C \\ T_F \end{bmatrix} \tag{10.12}$$

The electrical dynamics are significant at much higher frequencies than the mechanical dynamics. They can be removed from the equations by setting the derivative of z equal to zero and finding the steady-state value of z, which is

$$z = \frac{G}{R}\left(\frac{T_C}{k_T} + \frac{k_T}{R}\omega\right) \tag{10.13}$$

The reaction-wheel actuator can then be modeled with a single state equation. This is the equivalent of saying that $i_M = i_C$. The state equations are linear and do not include voltage and current limits. When implemented in code, this must be included. Fig. 10.4 shows a doublet (a positive then a negative torque) with and without current feedback. The effect of back-emf is evident.

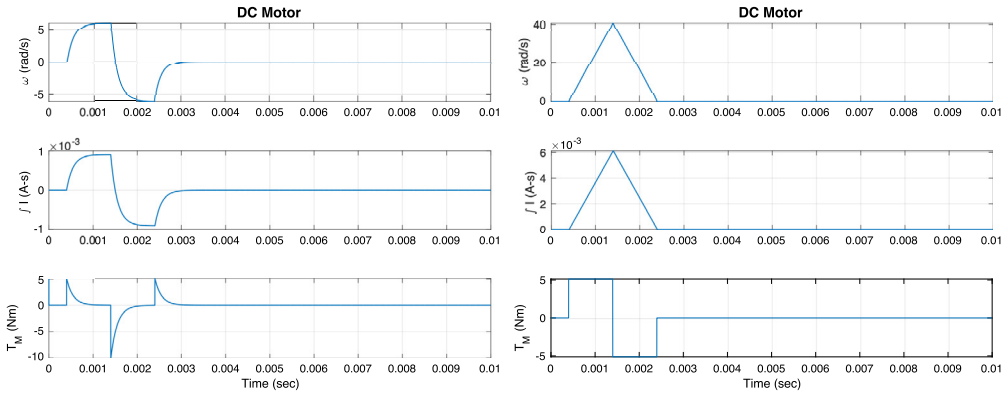


Figure 10.4 Response to a doublet with and without current feedback. The plot on the right shows current feedback.

10.4.5 Tachometer

Most spacecraft use permanent magnet motors. The rotor has poles made up of rare-Earth magnets. The stator currents are switched based on the angle of the magnets with respect to the stator. The angle is determined by Hall-effect sensors that give a pulse when the magnet pole passes over the switch. The time between Hall-sensor pulses can be used to determine the wheel rate. Motors can have many poles; For a 3-phase motor, only 3 Hall sensors are needed for commutation. However, up to 18 sensors are not unusual. Time is computed by running a counter and dumping the count when a

Hall sensor signals. If the motor is running fast, one count per cycle is sufficient. If it is running slowly, the count between each Hall sensor is output. If the motor is running sufficiently slow, there may not be an update available between samples. This presents problems for low-speed operation. One solution is to use current measurements to estimate the rate at low speed. Another is to put an angle encoder on the shaft.

10.4.6 Friction

Friction is an important consideration when modeling reaction wheels or any rotating or sliding component. The simplest model is that of Coulomb friction and viscous friction.

$$F = -\frac{v}{|v|} f_c - \sigma_v v \tag{10.14}$$

The first term is a constant force (or torque) that resists motion and is independent of speed. The second term is proportional to the speed (or angular rate). This model does not include stiction, also known as breakaway friction, which is the force (or torque) needed to get something moving. This can be modeled using the bristle-friction model [1]. This model adds a state that can be thought of as the bending of the bristles. The model is

$$F = \sigma_0 z + \sigma_1 \dot{z} + \sigma_v * v \tag{10.15}$$

where

$$\dot{z} = v - \frac{|v| z}{g(v)} \tag{10.16}$$

and

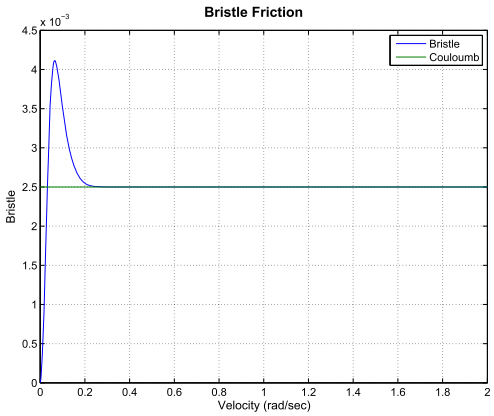
$$g(v) = \frac{f_C + (f_S - f_C) * e^{-\left(\frac{v_S}{v}\right)^2}}{\sigma_0} \tag{10.17}$$

Example 10.2 compares bristle friction with Coulomb friction. The parameters in the model are not readily available from vendors and must be obtained from testing.

10.4.7 Zero crossings

Near to zero wheel speed the stiction torque tries to keep the wheel at zero speed. With trapezoidal commutation using Hall sensors for rotor-angle measurements it is difficult to measure the wheel speed since you are not getting a steady stream of Hall-sensor pulses. One strategy is to bias up the wheels to avoid zero crossings. This requires more momentum storage but should improve the wheel life.

```
dT     = 0.0125;
nSim   = 1000;
fCoul  = 0.005/2;

dRHS.fStatic    = 0.005;
dRHS.fCoulomb   = fCoul;
dRHS.vStribeck  = 0.1;
dRHS.sigma0     = 1;
dRHS.sigma1     = 1e-4;
dRHS.sigma2     = 0;
dRHS.maxC       = 1/dT;

xPlot = zeros(2,nSim);
x     = 0;
v     = linspace(0,2,nSim);

for k = 1:nSim
  dRHS.v = v(k);
  x              = RK4( @FrictionBristle, x, dT,
        0, dRHS );
  [zDot, f, c]   = FrictionBristle( x, 0, dRHS );
  xPlot(:,k)     = [f;fCoul];
end

Plot2D( v, xPlot, 'Velocity (rad/sec)', 'Bristle'
      , 'Bristle Friction','lin')
legend({'Bristle' 'Couloumb'});
```



Example 10.2: Bristle friction and Coulomb friction.

10.4.8 Commutation

Commutation for motors with a trapezoidal drive uses Hall sensors (magnetic-field sensors). Six-state commutation directs current through two phases of the motor at a time, based on the Hall-sensor switches. The current switches when a Hall sensor changes state. The torque angle curve is not always at the ideal 90 degrees, resulting in torque ripple.

Table 10.2 shows the commutation pattern.

Table 10.2 Reaction-wheel commutation pattern.

Phase A	Phase B	Phase C
OFF	+	-
-	+	OFF
-	OFF	+
OFF	-	+
+	-	OFF
+	OFF	-

Fig. 10.5 shows the response to commutation. Note the torque noise.

10.4.9 Suspensions

Noise sources due to reaction wheels are:

1. Center-of-mass is not on the spin axis;

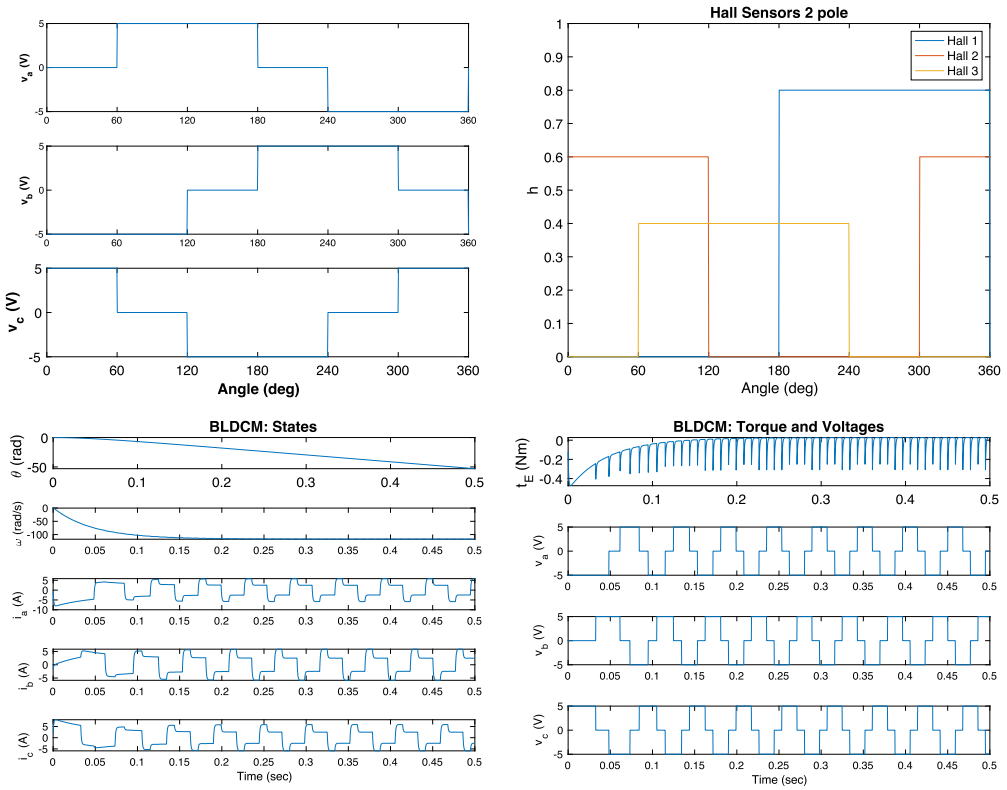


Figure 10.5 Response to commutation.

2. Nondiagonal inertia matrix;
3. Angular acceleration about z produces torque around all three axes;
4. Torque ripple;
5. Torque production is not linear due to commutation;
6. Cogging due to interaction of magnets with the stator steel;
7. Bearing noise.

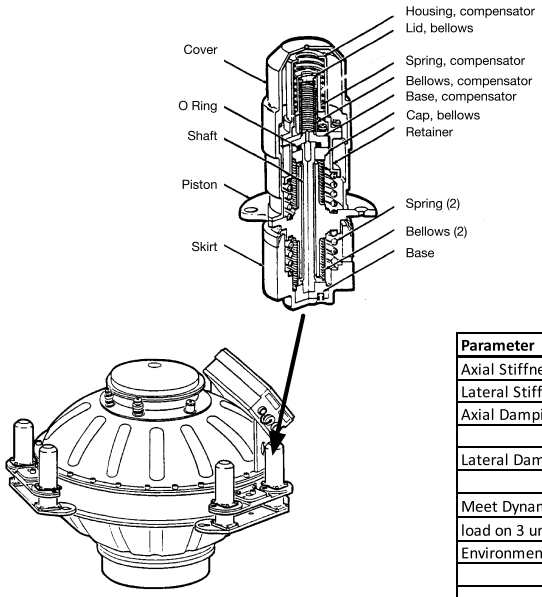
For precision pointers (James Webb, Hubble) suspensions are used.

Fig. 10.6 shows the suspension for the NASA Hubble Space Telescope wheels. The system uses a spring and a damper.

10.5. Control-moment gyro

10.5.1 Introduction

Reaction wheels are limited in torque capability and momentum-storage capability. When higher torques are needed, for example on a reconnaissance satellite that must



Parameter	Requirement
Axial Stiffness	670-910 lb/in
Lateral Stiffness	1380-1870 lb/in
Axial Damping	0.170-0.025
	(Q = 3 to 20)
Lateral Damping	0.170-0.025
	(Q = 3 to 20)
Meet Dynamic Properties with static	37.0 lbs axial
load on 3 units - 1G	102.0 lbs radial
Environment	Temp: -20 deg-F to 120 deg-F
	Pressure: 810 to 1e-13 Torr
	Accel: 9.2 g's for 1.5 min
	Random Vibe: 6.18 g RMS Overall
Life	Unit: 2 years ground
	5 year orbital
	Damping Element: 7 years
Weight	5.0 lbs max

Figure 10.6 Hubble Space Telescope suspension [2]. Image courtesy of NASA.

track a target on the Earth, control-moment gyros (CMGs) are often used. A CMG is a spinning wheel on a gimbaled platform. CMGs may have one or two gimbals. The wheel spin rate is kept constant with a tachometer loop. This degree-of-freedom is not used for control. The gimbal torquer motors are used for control.

10.5.2 Modeling

The CMG adds another body to the spacecraft with 2 or more degrees-of-freedom. Generally, the inner gimbal is limited to ± 90 degrees, while the outer gimbal can rotate 360 degrees. Power is supplied to the inner gimbal via slip rings. Any spacecraft with CMGs is a multibody spacecraft. For control-design purposes, however, it is useful to model the CMG drive as a high-bandwidth drive in which the gimbal rates achieve a commanded rate instantaneously. The output torque for one CMG is

$$T = \dot{B}H \tag{10.18}$$

where H is the rotor angular momentum and is constant. B is composed of two parts. One is constant and represents the alignment of the CMG base. The second includes the contributions due to the gimbal rotations. Assume that the rotor, in the CMG base frame, is aligned along the y-axis. Then, the torque equation becomes

$$T = Bh \begin{bmatrix} \cos\alpha & 0 \\ -\cos\beta\sin\alpha & -\sin\beta\cos\alpha \\ -\sin\beta\sin\alpha & \cos\beta\cos\alpha \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} \tag{10.19}$$

where B is the transformation matrix from the CMG base to the spacecraft frame. When α and β are zero, the CMG momentum for this particular selection of gimbal angles, in the CMG frame, is

$$H = \begin{bmatrix} 0 \\ h \\ 0 \end{bmatrix} \tag{10.20}$$

10.5.3 Torque distribution

Each double-gimbal CMG has two control degrees-of-freedom. At least two CMGs are need for 3-axis control, but even two will have four degrees-of-freedom. One way to distribute the torque demand is through a pseudoinverse law that penalizes the squared magnitude of the gimbal rates. If

$$T = A \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \end{bmatrix} \tag{10.21}$$

is the torque for one CMG then

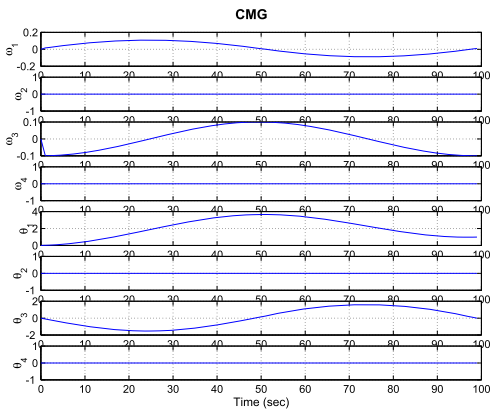
$$T = \begin{bmatrix} A_1 & A_2 \end{bmatrix} \begin{bmatrix} \dot{\alpha}_1 \\ \dot{\beta}_1 \\ \dot{\alpha}_2 \\ \dot{\beta}_3 \end{bmatrix} \tag{10.22}$$

is the torque for two CMGs. The gimbal rates are then

$$\begin{bmatrix} \dot{\alpha}_1 \\ \dot{\beta}_1 \\ \dot{\alpha}_2 \\ \dot{\beta}_3 \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \end{bmatrix}^T \left(\begin{bmatrix} A_1 & A_2 \end{bmatrix}^T \begin{bmatrix} A_1 & A_2 \end{bmatrix} \right)^{-1} T \tag{10.23}$$

This torque distribution law only requires the inverse of a 3-by-3 matrix. However, while it attempts to minimize the gimbal rates it does not try to keep the CMGs away from gimbal lock, a condition in which control is lost along one degree-of-freedom. If large CMG motion is expected, a more sophisticated approach will be required. Example 10.3 shows the gimbal response to an x and y disturbance torque.

```
h     = 10; n = 100; dT = 1.0;
b     = {eye(3) [0 1 0;-1 0 0;0 0 1]};
d     = linspace(0,2*pi,n);
t     = [sin(d)+0.1;cos(d);zeros(1,n)];
omega = zeros(4,n); theta = zeros(4,n);
a     = zeros(3,4);
for k = 2:n
  i = 1;
  for j = 1:2:4
    cA          = cos(theta(j,k)); sA = sin(
          theta(j,k));
    cB          = cos(theta(j+1,k)); sB = sin(
          theta(j+1,k));
    a(:,[j j+1]) = b{i}*[cA 0;-sB*sA -sB*cA;-sB*
          sA cB*cA]*h;
    i           = i + 1;
  end
  omega(:,k) = pinv(a)*t(:,k);
  theta(:,k) = theta(:,k-1) + dT*omega(:,k);
end
time = (0:(n-1))*dT;
Plot2D(time,[omega;theta],'Time (sec)',...
       {'\omega_1','\omega_2','\omega_3','\
            omega_4',...
        '\theta_1','\theta_2','\theta_3','\
            theta_4'},'CMG');
```



Example 10.3: Double-gimbal CMG.

10.5.4 Single-axis control-moment gyros

A single-axis control-moment gyro has a single-axis hinge between the gyroscope assembly and the spacecraft. This constraint implies that the momentum in the wheel will be seen as a momentum bias on the spacecraft. The control torque on the spacecraft is due to the precession of the wheel assembly.

$$T = B\Omega^{\times}h \tag{10.24}$$

where h is the wheel-momentum vector in the CMG frame, Ω is the precession rate vector, and B transforms from the base CMG frame to the body frame. The skew-symmetric matrix is

$$\Omega^{\times} = \begin{bmatrix} 0 & -\Omega_z & \Omega_y \\ \Omega_z & 0 & -\Omega_x \\ -\Omega_y & \Omega_x & 0 \end{bmatrix} \tag{10.25}$$

If we define that in the CMG frame, the nominal momentum vector is along z and the hinge axis is along x we get

$$T = B\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\delta & -\sin\delta \\ 0 & \sin\delta & \cos\delta \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\dot{\delta} \\ 0 & \dot{\delta} & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ h \end{bmatrix} = -B\begin{bmatrix} 0 \\ \cos\delta \\ \sin\delta \end{bmatrix} h\dot{\delta} \tag{10.26}$$

When $\delta = 0$

$$T = -B\begin{bmatrix} 0 \\ h\dot{\delta} \\ 0 \end{bmatrix} \tag{10.27}$$

By picking the appropriate B matrices we can generate 3-axis torques. However, as the gimbal angles change we will find that holes in the momentum envelope may appear.

10.6. Thrusters

10.6.1 Introduction

Small monopropellant and bipropellant thrusters are used for attitude control on many satellites. They are used for attitude control during orbit-change maneuvers or momentum unloading, as an alternative to magnetic torquers. Most small thrusters are on–off thrusters. That is, their valves have only two positions, on and off.

10.6.2 Pulsewidth modulation

An alternative is to control the pulse length of the thruster. If the control period is 1 second, for example, the average thrust can be halved if the pulsewidth is 0.5 s. The thrust does not change, but the average thrust over the period of control is halved. This is done at the expense of adding high-frequency control activity. For example, if a constant thrust of 1/2 the thruster thrust is desired, the actual thrust will be a square wave with a 50% duty cycle and a cycle period equal to the control period. Thus, the actual output of the thruster will have significant high-frequency harmonics and might excite vibratory modes in the system.

10.6.3 Minimum impulse bit

A major problem is that due to the dynamics of the valve, all thrusters have minimum impulse bits. Generally, 16 ms is the minimum on-time for a thruster. If the control period is 1 second, that gives an effective throttle ratio of 62.5:1. That is the equivalent of a 6-bit data word! In contrast, an audio CD uses 16-bit data words! This can be improved by using a multirate control system. The digital controller runs with a period

of 1 second, but the output is sampled less frequently. For example, the output might be sampled every 8 control periods. This improves the control resolution to 9 bits.

When designing a control system it is important to include the effects of undersampling. The easiest way to do this is to add a zero-order hold with a duration of the minimum impulse bit, followed by a delay of length equal to the period between output samples.

Fig. 10.7 shows how the impulse bit can vary with pulsewidth for a monopropellant thruster. The spike at the minimum pulsewidth increases the loop gain and can cause

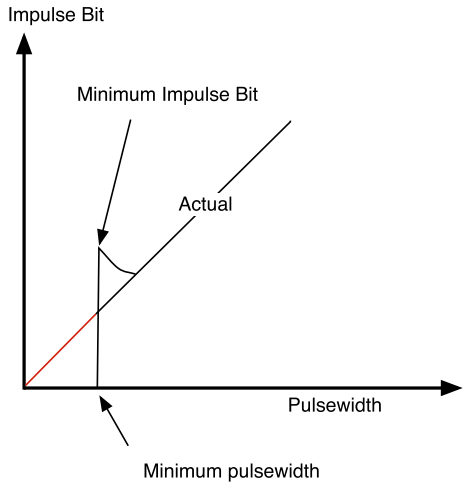


Figure 10.7 Impulse bit vs. pulsewidth.

problems with proportional-integral-derivative (PID) controllers as they are conditionally stable.

10.6.4 Time constants

Thrusters with solenoid valves can be modeled as first-order systems with separate rise and fall time constants. When fuel is flowing into the thruster the model is

$$\dot{u} + \frac{u}{\tau_f} = 1 \tag{10.28}$$

when no fuel is flowing the model is

$$\dot{u} + \frac{u}{\tau_f} = 1 \tag{10.29}$$

The thrust is

$$T = ku \tag{10.30}$$

where k is the maximum thrust given by

$$k = C_F A^* P \tag{10.31}$$

where P is the pressure in the rocket combustion chamber, C_F is the thrust coefficient and A^* is the throat area.

10.6.5 Fuel system

A typical fuel system is shown in Fig. 10.8. The system has two fuel volumes (that may consist of several tanks) with hydrazine fuel pressurized by helium. The systems

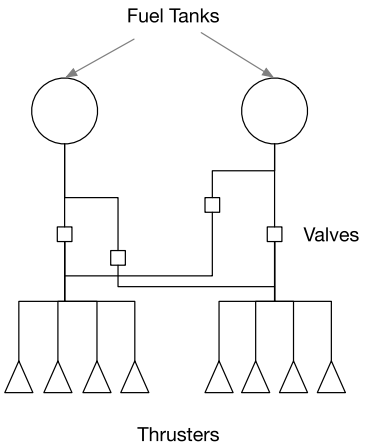


Figure 10.8 Tank geometry.

are cross-strapped (either set of thrusters can use either tank) with a total of four latch valves. Each half system is attached to six REAs.

If the system is unregulated, the pressure in each system is

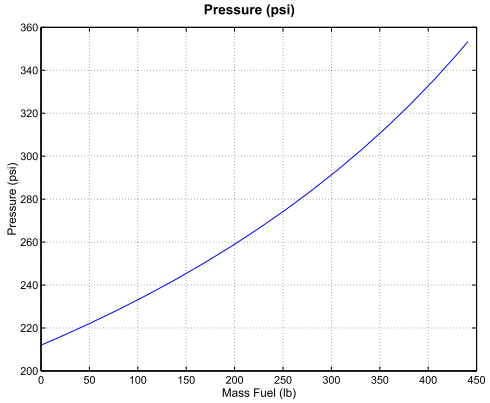
$$P = \frac{m_{\text{He}} R_{\text{He}} T}{V - \frac{m_{\text{Hydrazine}}}{\rho_{\text{Hydrazine}}}} \tag{10.32}$$

where m_{He} is the mass of helium, R_{He} is the gas constant of helium,

$$R_{\text{He}} = \frac{8.3}{0.004} \tag{10.33}$$

T is the gas temperature, V is the system volume (including any auxiliary pressurant tanks), $m_{\text{Hydrazine}}$ is the mass of the fuel, and $\rho_{\text{Hydrazine}}$ is the density of the fuel. The Demo in Example 10.4 shows a propellant system designed so that the pressure stays between 210 and 250 psi during the life of the mission.

```
Nm2ToPSI = 1.4504e-04;
KgToLbF  = 2.2046e+00;
molWt    = 0.004;
rPress   = 2.0786e+03;
mFuel    = 0:200;
rhoFuel  = 1000;
T        = 293;
vTank    = 0.5;
mPress   = 1.2;
p        = BloDown(mPress,rhoFuel,vTank,rPress,T,
           mFuel);

Plot2D(mFuel*KgToLbF,p*Nm2ToPSI,'Mass Fuel (lb)',
       'Pressure (psi)');
```



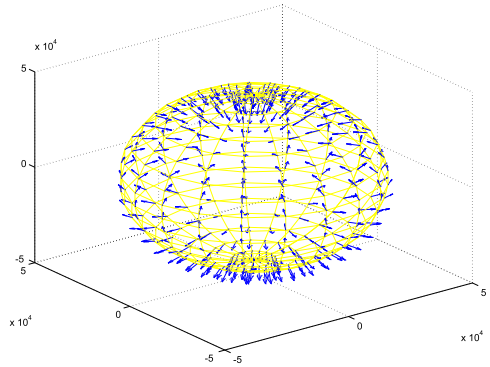
Example 10.4: Blowdown curve.

10.7. Magnetic torquers

10.7.1 The magnetic field

The magnetic-field vectors are shown in Example 10.5. This particular model represents the field as a tilted dipole, which is good for many low-Earth orbit applications.

```
% For a geo orbit
DrawFieldLines( 43200, 20, 'BDipole', 2451545.0 )
```



Example 10.5: Magnetic field at the geosynchronous-orbit altitude.

10.7.2 Magnetic torque

10.7.2.1 Torque production

Magnetic torquers are configured as either an air coil or a torquer bar. Torquers generate torque through interaction with the Earth's magnetic field

$$T = M^{\times} B \tag{10.34}$$

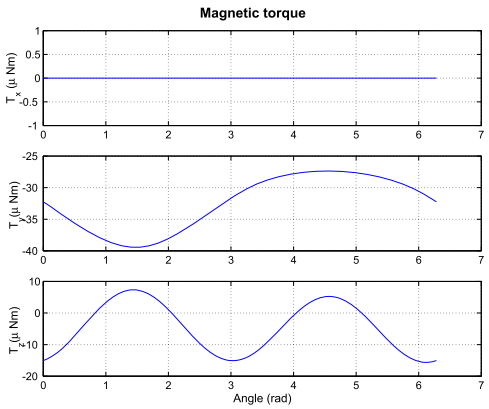
where B is the magnetic field and M is the dipole of the torquer. For an air coil, the dipole is

$$M = NIA \tag{10.35}$$

where N is the number of turns of wire, I is the current in the wire, and A is the area enclosed by the coil. Typically, the wire is wrapped around the corners of the spacecraft structure.

Example 10.6 shows the force vector generated by a 400 ATM^2 (Amp-turn-meters squared) magnetic torquer in any Earth-pointing spacecraft at geosynchronous orbit. The torquer is aligned with the x-axis, which is in the orbit plane. In geosynchronous orbit, the magnetic field is nearly normal to the orbit plane. In this example, the Sun is along the x-axis. The model accounts for solar activity, which is set to a maximum. At geosynchronous orbit, the magnetic field is strongly influenced by the Sun. The torque is in the body frame.

```
a  = linspace(0,2*pi);
c  = cos(a);
s  = sin(a);
n  = 100;

% Circular orbit
r  = 43200*[s;c;zeros(1,n)];
m  = [400;0;0];
jD = Date2JD([2010 3 21 0 0 0]);
t  = zeros(3,n);
for k = 1:n
   b       = [c(k) s(k) 0 ;-s(k) c(k) 0;0 0 1];
   t(:,k) = cross( m, b*BMF( r(:,k), jD, 3 ) );
end
Plot2D(a,t*1e6,'Angle (rad)',...
 {'T_x (\muNm)' 'T_y (\muNm)' 'T_z (\muNm)'},...
 'Magnetic Torque')
```



Example 10.6: Magnetic torque at geo.

Magnetic torquers cannot instantaneously produce a 3-axis torque. One approach is to compute the moments for the dipoles that fit the desired torque in a least-squares sense. Assume that there are n dipoles so that the dipole vector m is $n \times 1$. The dipole unit vector's matrix is u and is $3 \times n$. If the cost of using the torquers is the scalar c and the magnetic-field vector is b define

$$\gamma = u \times b \tag{10.36}$$

The scalar cost is

$$J = (\gamma m - t_D)^T (\gamma m - t_D) + cm^T m \tag{10.37}$$

Taking the partial derivative with respect to m we get the least-square dipole, which is

$$m = \left(\gamma^T \gamma + c\right)^{-1} \gamma t_D \tag{10.38}$$

where t_D is the torque demand.

10.7.2.2 Magnetic-torquer design

Torquer bars use magnetic steel to concentrate the flux, getting the same dipole in a smaller volume. However, the magnetic steel can saturate, limiting the dipole. In addition, the magnetic steel will always have some small residual dipole when off. The power consumed by a torquer is

$$P = \frac{V^2}{R} \tag{10.39}$$

where V is the voltage across the coil terminals and R is the total resistance of the coil. This is,

$$R = 2\sigma\pi r_w l_w \tag{10.40}$$

where r_w is the wire radius, σ is the resistivity of the wire, and the length of the wire is

$$l_w = 2\pi r_b N \tag{10.41}$$

where N is the total number of turns and r_b is the radius of the bar. The wire radius is only available as discrete American Wire Gauge (AWG) values. The dipole moment is

$$M = \pi r_b^2 N \frac{V}{R}\left(1 + \frac{\mu_r - 1}{1 + (\mu_r - 1)N_d}\right) \tag{10.42}$$

where μ_r is the relative permeability and the demagnetizing factor N_d is

$$N_d = \frac{4(\log\gamma - 1)}{\gamma^2 - 4\log\gamma} \tag{10.43}$$

where

$$\gamma = \frac{l_b}{r_b} \tag{10.44}$$

If we specify power and voltage, we determine R

$$P = IV \tag{10.45}$$

$$V = IR \tag{10.46}$$

$$R = \frac{V^2}{P} \tag{10.47}$$

This leads to the number of turns

$$N = \frac{R}{2\pi r_b \sigma \pi r_w^2} \tag{10.48}$$

where the denominator is the resistance per turn. The number of layers of wire is

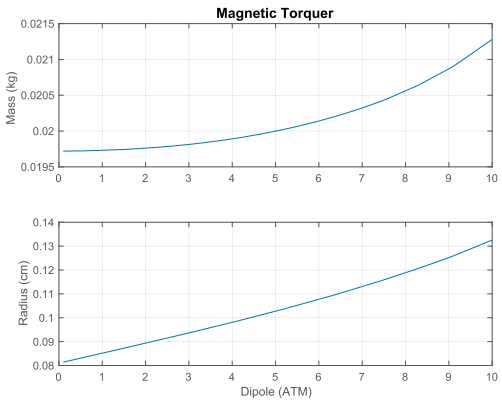
$$N_L = \frac{2Nr_w}{l_b} \tag{10.49}$$

The mass of the torquer is

$$m = \rho_b \pi r_b^2 l_b + 2\rho_w N \pi r_w^2 \pi r_b \tag{10.50}$$

where ρ_b is the density of the bar and ρ_w is the density of the wire. Example 10.7 shows how the mass, dipole, and radius of a torquer varies with terminal voltage. The radius increases by steps because the number of winding layers is an integer. Higher voltages produce more torque per unit mass. This torquer uses Hiperco-A, which has a relative permeability of 15 000. Relative permeabilities above 2000 do not change the mass.

```
d.aWG                = 32;
d.power              = 4;
d.v                  = 12;
d.dipole             = 4;
d.densityConductor   = 8960;
d.conductivity       = 59.6e6;
d.densitySteel       = 7650;
d.muR                = 2000;
d.l                  = 0.22;
dipole               = logspace(-1,1);
n                    = length(dipole);
mass                 = zeros(1,n);
radius               = zeros(1,n);
for k = 1:n
  d.dipole  = dipole(k);
  d         = MagneticTorquerDesign( d );
  mass(k)   = d.m;
  radius(k)     = d.rTorquer;
end
yL  = {'Mass (kg)', 'Radius (cm)'};

Plot2D(dipole,[mass;radius*100],'Dipole (ATM)',yL
      ,'Magnetic Torquer');
```



Example 10.7: Magnetic-torquer mass and radius vs. voltage.

10.8. Solenoids

10.8.1 Introduction

Solenoids are electromechanical devices that are used in many types of spacecraft systems. This section derives the equations of motion for a dual-coil solenoid. Besides giving models of solenoids, these equations are similar to those used to model rotary and linear motors.

10.8.2 Derivation of the equations of motion for a dual-coil solenoid

The solenoid is illustrated in Fig. 10.9. The first step is to find the magnetic fields that

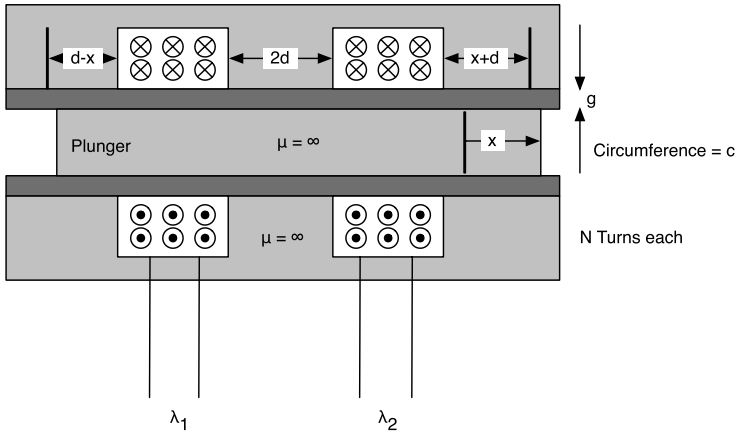


Figure 10.9 Dual-coil solenoid.

are significant only in the sleeve where the permeability is that of free space. In both the plunger and the enclosure the permeability is assumed to be infinite; hence the fields are nearly zero. The magnetic fields are found from the equation

$$-\oint_C H \cdot dl = \int_S J \cdot nda \tag{10.51}$$

This equation means that if you integrate the magnetic fields on a closed path around a surface *S*, the integral will equal the sum of all of the currents that flow normal to the surface and are within the path. In the above drawing, there are three possible paths. One surrounds just coil 1, one surrounds just coil 2, and one surrounds both coils. There are also three regions in which the magnetic field can be found. One is to the left of coil 1, one is between coil 1 and 2 and the third is to the right of coil 3. Only two paths are required to relate the fields, see Fig. 10.10. The two integrals are

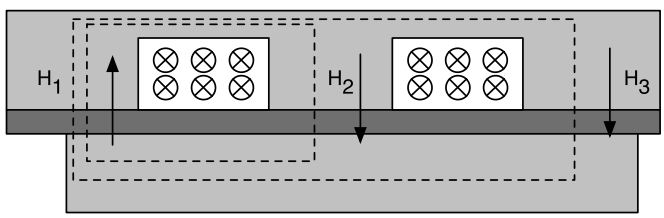


Figure 10.10 Contours for integrating the magnetic field.

$$gH_1 + gH_2 = Ni_1 \tag{10.52}$$

$$gH_1 + gH_3 = N(i_1 + i_2) \tag{10.53}$$

A third relationship is needed to solve for the fields. The surface integral of the magnetic flux over the region in which there are no sources must be zero, or

$$-\oint_S B \cdot nda = 0 \tag{10.54}$$

This surface integral is over a volume, unlike the one in Eq. (14.6). This results in

$$\mu_o c((d-x)H_1 - 2dH_2 - (d+x)H_3) = 0 \tag{10.55}$$

Substituting for H_2 and H_3 gives

$$H_1 = \frac{N}{4g}\left(2i_1 + \left(1 + x/d\right)(i_1 + i_2)\right) \tag{10.56}$$

$$H_2 = \frac{N}{4g}\left(2i_2 + \left(1 - x/d\right)(i_1 + i_2)\right) \tag{10.57}$$

To find the flux linked by each of the coils, it is necessary to find the flux through the volume within each of the coils. For each coil this is

$$\phi_1 = \mu_o cd\left(1 - x/d\right)H_1 \;\; \phi_2 = \mu_o cd\left(1 + x/d\right)H_3 \tag{10.58}$$

Each flux is linked N times by each coil, so that total flux linked becomes

$$\lambda_1 = \frac{\mu_o N^2 cd}{4g}\left(\left(3 - 2\frac{x}{d} - \frac{x^2}{d^2}\right)i_1 + \left(1 - \frac{x^2}{d^2}\right)i_2\right) \tag{10.59}$$

$$\lambda_2 = \frac{\mu_o N^2 cd}{4g}\left(\left(1 - \frac{x^2}{d^2}\right)i_1 + \left(1 + 2\frac{x}{d} - \frac{x^2}{d^2}\right)i_2\right) \tag{10.60}$$

The inductances are

$$L_{11} = L_o\left(3 - 2\frac{x}{d} - \frac{x^2}{d^2}\right) \tag{10.61}$$

$$L_{22} = L_o\left(3 + 2\frac{x}{d} - \frac{x^2}{d^2}\right) \tag{10.62}$$

$$L_{12} = L_o\left(1 - \frac{x^2}{d^2}\right) \tag{10.63}$$

$$L_0 = \frac{\mu_o N^2 cd}{4g} \tag{10.64}$$

The terminal voltages are

$$v = iR + \frac{d\lambda}{dt} \tag{10.65}$$

or

$$v_1 = i_1 R + L_{11}\frac{di_1}{dt} + L_{12}\frac{di_2}{dt} - \frac{2L_o}{d}\left(\left(1+\frac{x}{d}\right)i_1 + \left(\frac{x}{d}\right)i_2\right)\frac{dx}{dt} \tag{10.66}$$

$$v_2 = i_2 R + L_{121}\frac{di_1}{dt} + L_{22}\frac{di_2}{dt} - \frac{2L_o}{d}\left(\left(\frac{x}{d}\right)i_1 + \left(\frac{x}{d}-1\right)i_2\right)\frac{dx}{dt} \tag{10.67}$$

The electrical force is

$$f_e = \frac{\partial W'_m}{\partial x} \tag{10.68}$$

where W'_m is the coenergy. For this example,

$$W'_m = \int_0^{i_1} \lambda_1 di_1 + \int_0^{i_2} \lambda_2 di_2 \tag{10.69}$$

this becomes

$$W'_m = \frac{1}{2}L_{11}i_1^2 + 2L_{12}i_1 i_2 + \frac{1}{2}L_{22}i_2^2 \tag{10.70}$$

and the force on the plunger is

$$f_e = \frac{1}{2}\frac{\partial L_{11}}{\partial x}i_1^2 + 2L_{12}\frac{\partial L_{12}}{\partial x}i_1 i_2 + \frac{1}{2}\frac{\partial L_{22}}{\partial x}i_2^2 \tag{10.71}$$

or

$$f_e = \frac{L_o}{d}\left(\left(1-\frac{x}{d}\right)i_2^2 - 4\left(\frac{x}{d}\right)i_1 i_2 - \left(1+\frac{x}{d}\right)i_1^2\right) \tag{10.72}$$

$$f_e = \frac{L_o}{d}\left(i_2^2 - i_1^2 - \frac{x}{d}(i_2^2 + i_1^2 + 4i_1 i_2)\right) \tag{10.73}$$

The equation of motion for the mechanical part of the system is

$$M\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx + b\frac{dx/dt}{|dx/dt|} = \frac{L_o}{d}\left(i_2^2 - i_1^2 - \frac{x}{d}(i_2^2 + i_1^2 + 4i_1 i_2)\right) \tag{10.74}$$

The first term is the inertial acceleration, the second is viscous damping, the third is linear spring stiffness, and the fourth is Coulomb friction. If the currents are balanced and constant the last term in the coupling term provides stiffness to the system. The mechanical equations are linear in x and the electrical equations are linear in i, and the coupled equations are nonlinear in both x and i.

10.8.3 Derivation of the equations of motion for a single-coil solenoid

The solenoid is illustrated in Fig. 10.11. The first step is to find the H fields. The H

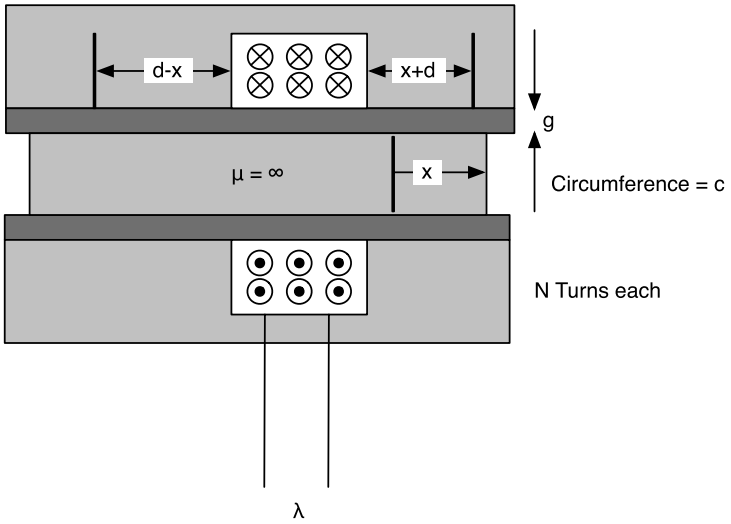


Figure 10.11 Single-coil solenoid.

fields are significant only in the sleeve where the permeability is that of free space. In both the plunger and the enclosure, the permeability is assumed to be infinite; hence the fields are nearly zero. The magnetic fields are found from the equation

$$\oint_C H \cdot dl = \int_S J \cdot nda \tag{10.75}$$

This equation means that if you integrate the magnetic fields on a closed path around a surface S, the integral will equal the sum of all of the currents that flow normal to the surface and are within the path. In the above drawing, there are three possible paths. One surrounds just coil 1, one surrounds just coil 2, and one surrounds both coils. There are also three regions in which the magnetic field can be found. One is to the left of coil 1, one is between coil 1 and 2, and the third is to the right of coil 3. Only two paths are required to relate the fields, see Fig. 10.12. The two integrals are

$$gH_1 + gH_2 = Ni \tag{10.76}$$

A third relationship is needed to solve for the fields. The surface integral of the magnetic flux over a region in which there are no sources must be zero, or

$$\oint_S B \cdot nda = 0 \tag{10.77}$$

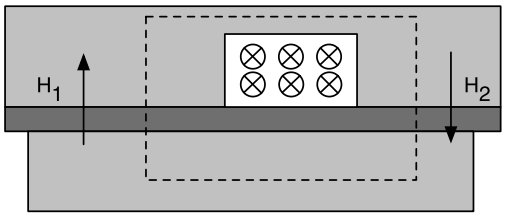


Figure 10.12 Contours for integrating the magnetic field.

This surface integral is over a volume. This results in

$$\mu_0 c((d-x)H_1 - (d+x)H_2) = 0 \tag{10.78}$$

Substituting for H_2 gives

$$H_2 = \frac{N}{g}\left(1 - \frac{x}{d}\right) i \tag{10.79}$$

To find the flux linked by the coil, it is necessary to find the flux through the volume within the coil. This is

$$\phi = \mu_o cd\left(1 + \frac{x}{d}\right) H_2 \tag{10.80}$$

Each flux is linked N times by the coil so the total flux linked becomes

$$\lambda = \frac{\mu_o cdN^2}{g}\left(1 - \frac{x^2}{d^2}\right) i \tag{10.81}$$

The inductances are

$$L = L_o\left(1 - \frac{x^2}{d^2}\right) \tag{10.82}$$

$$L_o = \frac{\mu_o cdN^2}{g} \tag{10.83}$$

The terminal voltage is

$$v = iR + \frac{d\lambda}{dt} \tag{10.84}$$

or

$$v = iR + L\frac{di}{dt} - \frac{2L_o xi}{d^2}\frac{d\lambda}{dt} \tag{10.85}$$

The electrical force is

$$f_e = \frac{\partial W'_m}{\partial x}. \tag{10.86}$$

For this example,

$$\partial W'_m = \int_o^i \lambda i. \tag{10.87}$$

this becomes

$$\partial W'_m = \frac{1}{2} L i^2 \tag{10.88}$$

and the force on the plunger is

$$f_e = \frac{1}{2} \frac{\partial L}{\partial x} i^2 \tag{10.89}$$

or

$$f_e = -\frac{L_o x i^2}{d^2} \tag{10.90}$$

The equations of motion for the system are

$$M\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx + b\frac{dx/dt}{|dx/dt|} = -\frac{L_o x i^2}{d^2} v = iR + L\frac{di}{dt} - \frac{2L_o x i}{d^2} dx dt \tag{10.91}$$

The first term is inertial acceleration, the second is linear spring stiffness, and the third is Coulomb friction. When the current is off and motion has damped the damper will sit at $-x_0$. When the current is added, the equation of motion will be

$$M\frac{d^2x}{dt^2} = \frac{L_o x i^2}{d^2} \tag{10.92}$$

which will push the solenoid in the $+x$ direction. This setup will work well as a valve driver, see Fig. 10.13.

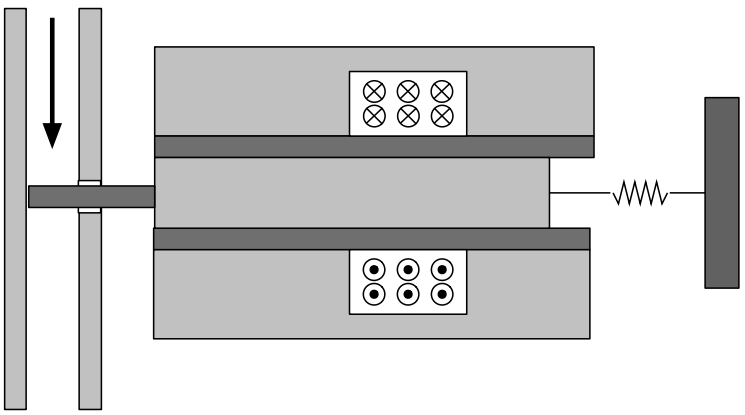


Figure 10.13 Valve driver.

10.9. Stepping motor

Stepping motors can be of many different types including variable reluctance, permanent magnet, hybrid (a combination of variable reluctance and permanent magnet), and claw-tooth permanent magnet, among others. A permanent-magnet motor was chosen as an example to demonstrate how the dynamics of the motor may be handled in models.

A two-phase, permanent-magnet stepping motor consists of a slotted stator with two phases and a permanent-magnet rotor. By energizing the phases of the stator the rotor can be caused to move one step.

The mathematical model of the stepping motor in *ab* (phase) coordinates is

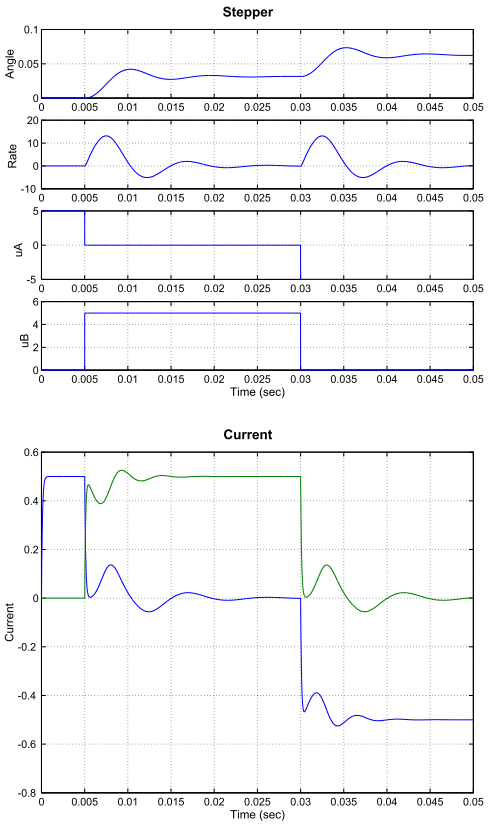
$$\begin{aligned}
\frac{di_a}{dt} &= \frac{1}{L}\left(u_a - Ri_a + K_m\omega s\right) \\
\frac{di_b}{dt} &= \frac{1}{L}\left(u_b - Ri_b - K_m\omega c\right) \\
\frac{d\omega}{dt} &= \frac{K_m}{J}\left(i_b c - i_a s - \frac{B}{K_m}\omega\right) \\
\frac{d\theta}{dt} &= \omega \\
c &= \cos(N_r\theta) \\
s &= \sin(N_r\theta)
\end{aligned} \tag{10.93}$$

The first two equations are the current equations for phases *a* and *b*. R is the resistance of the phase winding, K_m is the motor-torque constant, N_r is the number of rotor teeth, J is the rotor inertia, B is the viscous damping coefficient, u_a and u_b are the phase voltages, and i_a and i_b are the phase currents.

The results from a stepping-motor simulation are shown in Example 10.8. Note that the period of the simulation is only 50 ms. In that time the motor steps twice. Stepping is accomplished by setting the phase voltages to ±5 V.

The stepping-motor response looks like a moderately damped second-order system. As can be seen from the results, simulating the stepping motor would require 1-ms integration time steps. The dynamics of the stepping motor are far outside the bandwidth of the controller and therefore the approach will be to ignore the transients and assume that when the stepping motor is stepping the mass is moving at the desired velocity all the time. The model is then derived assuming that the mass velocity is constrained. When the mass velocity changes, the core-body rates are changed instantaneously so that angular momentum is conserved.

```
tSim         = 0.05;
dT           = 0.000005;
nSim         = tSim/dT;
dS.l         = 0.0011; % Inductance
dS.nR        = 50;     % Number of teeth
dS.r         = 10;     % Resistance
dS.kM        = 0.113;  % Motor torque
dS.kD        = 0;      % Detente torque
dS.tM        = 0.0;    % Mechanical shaft torques
dS.coulomb   = 0.0;
dS.inertia   = 5.7e-6;
dS.damping   = 0.001;
dS.mode      = 'ab';

xPlot        = zeros(6,nSim);
tPlot        = zeros(1,nSim);

x            = [0;0;0;0];
t            = 0;
dS.uA        = 5;
dS.uB        = 0;

for k = 1:nSim
  if( t >= 0.005 && t <= 0.03 )
    dS.uA =  0;
    dS.uB =  5;
  elseif( t > 0.03 )
    dS.uA = -5;
    dS.uB =  0;
  end

  xPlot(:,k) = [x;dS.uA;dS.uB];
  tPlot(k)   = t;
  x          = RK4('StepSim',x,dT,t,dS);
  t          = t + dT;
end
g =  [1 2 5 6];
Plot2D( tPlot, xPlot(g,:), 'Time (sec)',...
{'Angle' 'Rate', 'uA' 'uB'},'Stepper')
Plot2D( tPlot, xPlot(3:4,:), 'Time (sec)','
       Current','Current')
```



Example 10.8: Stepping motor.

10.10. Dampers

Dampers are used to dissipate energy in a spacecraft. Fuel slosh can act as a damper. Mechanical joints also provide damping. For example, deployable solar panels have numerous joints and they all act as dampers.

Appendix L gives the theory about magnetoresistive dampers, a particularly simple type of damper. Ball-in-tube dampers are also used on spacecraft.

References

[1] C.C. de Wit, H. Olsson, K.J. Astrom, P. Lischinksy, A new model for control of systems with friction, IEEE Transactions on Automatic Control 40 (3) (March 1995) 419–425.

[2] M. Hasha, Passive isolation/damping system for the Hubble Space Telescope reaction wheels, in: The 21st Aerospace Mechanisms Symposium, NASA-Lyndon B. Johnson Space Center, May 1987.

CHAPTER 11

Sensors

11.1. Space story

On the BSat launch in Japan, we were doing spin-axis attitude determination. Every time the spinning spacecraft saw the Sun it would measure the angle. We got files of angles. During transfer orbit, the spacecraft spin-axis was pointed in different directions. At one point we noticed a jump in Sun angle. The way the sensor worked was it had Sun detectors in a row and as the Sun moved along the row it would illuminate a different sensor giving an angle. Our customer ACS engineer realized that one of the detectors must be damaged so that its angle was never read. This was easily accommodated in the spin-axis attitude determination process, which was ground based.

11.2. Introduction

The performance of a spacecraft-control system is limited by the performance of its sensors and actuators. This chapter discusses the types of sensors that are used in spacecraft-control systems. Many of the sensors are used for determining the attitude or attitude rates of the spacecraft. Others are used for determining the relative orientation or position of components on a spacecraft.

11.3. Types of sensors

Table 11.1 lists the classes of sensors used in spacecraft. Within each class, there may be many types of sensors.

Table 11.1 Classes of sensors.

Sensor	Description	Application
Accelerometer	Measures acceleration on an element.	Inertial acceleration measurements, vibration sensing, nutation measurements.
Angle encoder	Measures either the absolute or relative angle of a shaft.	Angle measurements or angular-rate measurements.
Earth or planetary sensor	Measures the location of the sensor boresight relative to the nadir vector. Does not require the spacecraft to be rotating.	Orientation with respect to a planet.

continued on next page

https://doi.org/10.1016/B978-0-32-399915-1.00022-X

Table 11.1 (*continued*)

Sensor	Description	Application
Gyro	Any device that measures angular rates.	Rotation angle and rate measurements.
Hall effect	Measures the change in a magnetic field.	Used to determine rotor position in a motor.
Horizon sensor	Detects the Earth's horizon. Requires spacecraft rotation.	Attitude determination for spinning spacecraft. Planetary width for navigation.
Magnetometer	Measures the vector magnitude of a magnetic field.	Attitude determination. Magnetic control.
Potentiometer	A variable-resistance device in which the resistance varies as it is rotated.	Measurement of angles between rotating spacecraft components.
Rangefinder	Determines the distance between two objects that are not connected.	Docking, orbit maneuvers.
Star mapper	Measures the times of the crossing of a star across a detector.	Inertial attitude determination.
Star tracker	Measures the position of a star image in the focal plane of the sensor.	Inertial attitude determination.
Sun sensor	Either a single-axis or two-axis measurement of the angle of the Sun with respect to the boresight of the sensor.	Attitude determination, in particular yaw-attitude determination. Also used for Sun-safe modes.

11.4. Planetary optical sensors

This section covers optical sensors that image a planet or other distributed body.

11.4.1 Horizon sensors

A horizon sensor uses the spacecraft to scan the horizon of a celestial body. The measurement geometry for the Earth is shown on the left in Fig. 11.1. The horizon sensor has a very narrow field of view and responds to the scanning of the Earth as shown below. The ideal Earth pulse, which is the time history of the sensor that shows the heat from the Earth, is shown on the right of Fig. 11.1.

Several problems can arise. If a bright object is near the main object, the scan will be extended, causing the sensor to think it is closer to the equator of the body than it is. A second problem can occur if there is a dim spot on the scanned body. The sensor may then decide the edge of the body has been detected. The shortened chord makes it think the scan is nearer the pole.

Generally, the main problem is detecting the edge. The Earth's CO_2 layer is about 40 km above the surface of the Earth. The layer is not constant and varies with the season of the part of the Earth being scanned. For "precise" attitude determination using this

kind of sensor, it is necessary to maintain correction factors for seasonal variations in the edge. The CO_2 layer is used because, despite these issues, it is more reliable than using the visible bands.

Horizon sensors are used for attitude determination on spinning or dual-spin spacecraft. Many spacecraft spin in transfer orbit and a horizon sensor, in conjunction with a single-axis Sun sensor, can provide complete spin-axis orientation information. Another example is the GGS Polar spacecraft that used a horizon sensor on the spun point for pitch-orientation measurements during its mission.

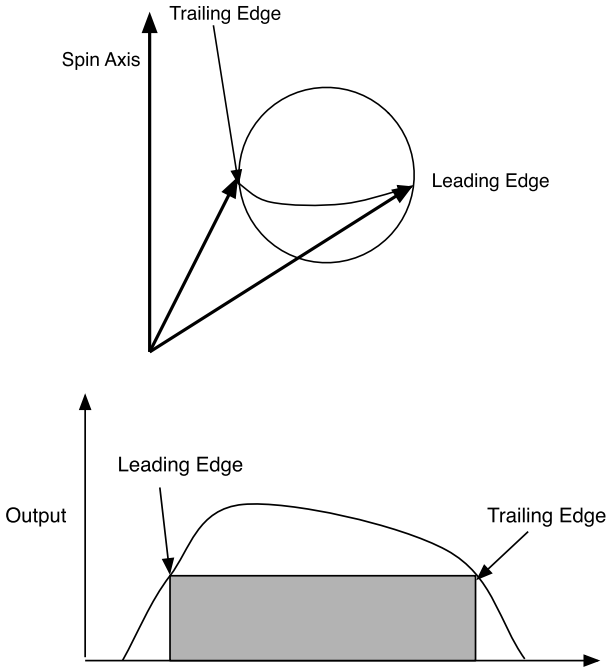


Figure 11.1 Horizon sensor measurement geometry and pulse.

11.4.2 Earth and planetary sensors

Earth and planetary sensors are sensors that detect the edge of the planet without requiring the spacecraft to spin. These also use an imager that sees the CO_2 band. There are three types: torsion-bar scanning, array, and conical scanning. The first uses an oscillating mirror to scan the Earth. The latter is similar to a horizon sensor. These sensors use the chordwidths to determine the roll attitude and the time from scan center to scan edge to determine pitch.

Scanning sensors measure roll by measuring the planetary chord at a specific cant angle and differencing the Earth-chord measurements the measurements to get an ap-

proximation to roll. The measured roll angle for a scanning sensor is

$$\theta_{meas} = 2\left[\cos^{-1}\left(\frac{\cos\rho - \sin\delta\sin\theta}{\cos\delta\cos\theta}\right) - \cos^{-1}\left(\frac{\cos\rho + \sin\delta\sin\theta}{\cos\delta\cos\theta}\right)\right] \tag{11.1}$$

where δ is the angle of the scan as measured from the plane normal to the sensor scan axis positive about the roll axis, ρ is the angular radius of the Earth, and θ is roll. This relationship can be proven by assuming that

$$\gamma + \theta = \cos^{-1}\left(\frac{\cos\rho - \sin\delta\sin\theta}{\cos\delta\cos\theta}\right) \tag{11.2}$$

Taking the cosine of both sides and assuming that θ is small

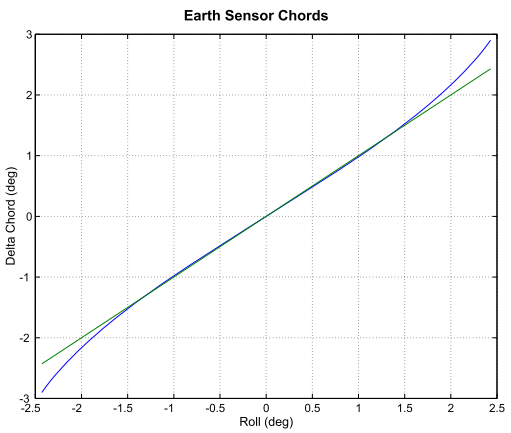
$$\cos\gamma - \theta\sin\gamma = \frac{\cos\rho}{\cos\delta} - \theta\frac{\sin\delta}{\cos\delta} \tag{11.3}$$

This is true if $\delta \approx \rho$ so that

$$\theta_{meas} \approx \theta \tag{11.4}$$

This is shown in Example 11.1. The approximation is excellent except at the very ends of the range.

```
1 rho = asin(6378/42167);
2 delta = 6*pi/180;
3 roll = 0.9*linspace(delta-rho,rho-delta);
4 ESAChord( rho, delta, roll )
5 PrintFig(1,1,1,'ESAChord')
```



Example 11.1: Earth-sensor chord.

A major advantage of these sensors is that they directly give roll and pitch outputs. The range of this type of sensor can be enhanced by defining a standard chord, a moving average of the North and South chords. When both chords are on the Earth

$$\theta = \frac{1}{2}(\Omega_s - \Omega_n) \tag{11.5}$$

Then, when the South chord goes off the Earth roll can be approximated by

$$\theta = \Omega_a - \Omega_n \tag{11.6}$$

or when the North chord goes off the Earth

$$\theta = \Omega_s - \Omega_a \tag{11.7}$$

The sensor, with this simple logic, provides the correct sign while at least one chord is on the Earth, as shown in Example 11.2. There is also a roll-reversal region that can prove dangerous if a controller tries to use it to correct roll not knowing that the sign is reversed.

The static Earth sensor has sets of thermopiles arranged in a circle about the bore-sight. Typically each set has three thermopiles. One looks at the Earth all of the time, the second straddles the Earth, and the third looks at "cold" space. The former and latter are used to calibrate the straddling sensor. The calibrated temperatures of the straddling sensors are used to compute roll and pitch.

The second Earth sensor detects the edges with a fixed set of sensors in a circular array. This type can use solid sensing elements that ride on the edge of the Earth or use an array to determine the edge. The conical scanning Earth sensor has a motor that rotates a mirror or prism that sweeps the field of view of the sensing element across the Earth, see Fig. 11.2.

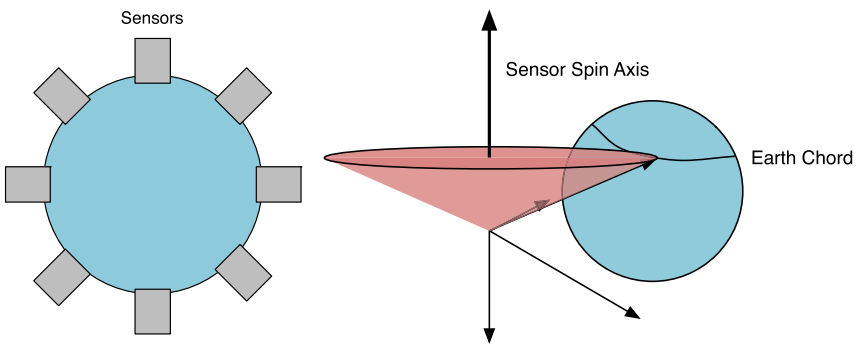
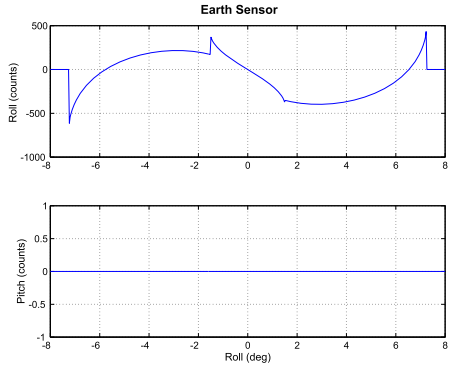


Figure 11.2 Static Earth sensor, left, and conical scanning Earth sensor on the right.

Pitch is found by measuring the center of the chord with respect to an internal reference and roll is found from the length of the chord. If the scan angle is set properly the scan will traverse the northern or southern hemisphere of the Earth resulting in a one-to-one relationship between chordwidth and attitude. This will not be the case for large attitude errors.

```
EarthSensorScanning('initialize', [], 42167 );
EarthSensorScanning( 'set noise', 0, 42167 );

clear d
d.rECI = [0;0;-42167];
a      = linspace(-8,8,500);
x      = zeros(2,500);
for k  = 1:length(a)
  d.qECIToBody = [CosD(a(k));SinD(a(k));0;0];
  EarthSensorScanning( 'update', d );
  t = EarthSensorScanning( 'get output', d );
  x(1,k) = t(1);
  x(2,k) = t(2);
end
yL = {'Roll (counts)','Pitch (counts)'};
Plot2D( a, x, 'Roll (deg)', yL, 'Earth Sensor')

%% Scanning Earth sensor function
function t = EarthSensorScanning( action, d,
      rStandard )

persistent s p

switch action

  case 'initialize'
    if( nargin < 2 )
      d = [];
    end

    if( isempty(d) )
      p = Default;
    else
      p  = d;
    end
    p.x              = Unit( Cross( p.uScan, p.
          boresight ) );
    s.scan           = [1 1];
    s.rollCounts     = 0;
    s.pitchCounts    = 0;
    s.valid          = 0;
    p.earthRadius    = 6378 + 40; % CO2
    p.cant           = pi/2 - p.scanAngle;
    s.failure        = 0;
    rho              = asin( p.earthRadius /
          rStandard );
    chord            = Chordwidth( p.uScan, rho, p
          .cant, [0;0;-1] );
    s.standardChord = 0.5*(chord(1)+chord(2));

  case 'set noise'
    p.sigma1Chord = d;

  case 'update'
    rho    =  asin( p.earthRadius / Mag( d.rECI )
          );
    uNadir = -QForm( d.qECIToBody, Unit( d.rECI )
          );
    noise  =  p.sigma1Chord*randn(1,2);
    chord  =  Chordwidth( p.uScan, rho, p.cant,
          uNadir ) + noise;

    % Pitch measurement
    pitch  = atan2( uNadir'*p.x, uNadir'*p.
          boresight );

    % Process the chordwidth data
    s.valid = 1;
    if( chord(1) > 0 && chord(2) > 0 & s.scan(1)
          && s.scan(2) )
      s.standardChord = s.standardChord/2 + (
            chord(2) + chord(1))/4;
      roll            = (chord(2) - chord(1))/2;
      pitch           = pitch + 0.5*(noise(1) +
            noise(2));
    elseif( chord(2) > 0 && s.scan(2) && s.
          standardChord > 0 )
      roll            = chord(2) - s.
            standardChord;
      pitch           = pitch + noise(2);
    elseif( chord(1) > 0 && s.scan(1) && s.
          standardChord > 0 )
      roll            = s.standardChord - chord
            (1);
      pitch           = pitch + noise(1);
    else
      roll            = 0;
      s.valid         = 0;
    end

    % Quantize
    if( s.failure )
      roll  = 0;
      pitch = 0;
    end
    s.rollCounts  = round(  roll/p.quantization )
          ;
    s.pitchCounts = round( pitch/p.quantization )
          ;

  case 'get output'
    t = [s.rollCounts; s.pitchCounts; round( s.
          standardChord/p.quantization ); s.
          valid];

end

end

%%  Chordwidth calculation
function w = Chordwidth( u, rho, cant, nadir )

cNadir = u'*nadir;
sNadir = sqrt( (1-cNadir)*(1+cNadir) );

arg    = (cos(rho)-cNadir*cos(cant))./(sNadir*sin
      (cant));

k      = find(abs(arg) < 1 );
w      = [0 0];
w(k)   = acos(arg(k));

end

%% Chordwidth calculation
function p = Default

p.scanAngle        = 5.72*(pi/180)*[-1 1];
p.uScan            = [0;1;0];
p.quantization     = 0.01*pi/180;
p.sigma1Chord      = 0.001*pi/180;
p.boresight        = [0;0;1];

end
```



Example 11.2: Scanning Earth sensor with standard chord.

11.4.3 Scanning Earth sensor

One type of Earth sensor is a scanning sensor with two scanning elements. The Earth edges are measured directly in degrees relative to the sensor null. The sensor calculates the actual chordwidth that will be seen by the North and South scans, and the dihedral angle, or angular separation of the sensor null from the center of the Earth. The dihedral angle is essentially the same as the pitch angle. The geometry that is the basis for the calculations is shown in Fig. 11.3. The positions of the scans are defined by their cant angles or angular separation from the sensor spin axis.

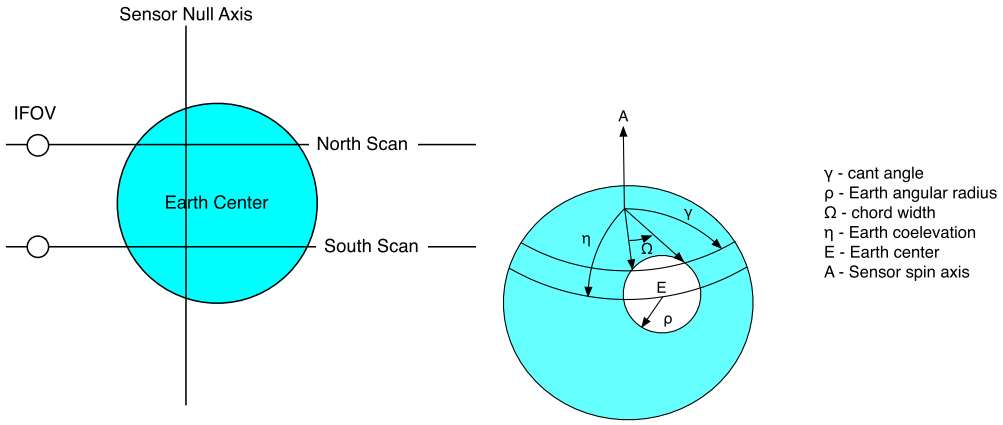


Figure 11.3 Diagram of the Earth-sensor scanning and chordwidth geometry.

$$\Omega = 2\cos^{-1}\left(\frac{\cos\rho - \cos\eta\cos\gamma}{\sin\eta\sin\gamma}\right) \tag{11.8}$$

$$\cos\eta = \vec{A}\cdot\vec{E} \tag{11.9}$$

The sensor scanning uses back and forth scans to calculate each roll and pitch measurement. Fig. 11.4 shows the measured quantities.

Roll and pitch are calculated from these quantities according to:

$$\text{pitch} = \frac{1}{8}\sum_{j=1}^{2}(-L_{j1} + L_{j2} + L_{j3} - L_{j4}) \tag{11.10}$$

$$\text{roll} = \frac{1}{4}\sum_{j=1}^{4}(L_{2i} - L_{1i}) \tag{11.11}$$

The sensor calculates L_{ji} from the actual chordwidth and dihedral angles, and then computes roll and pitch in the same manner as the sensor, using integer math.

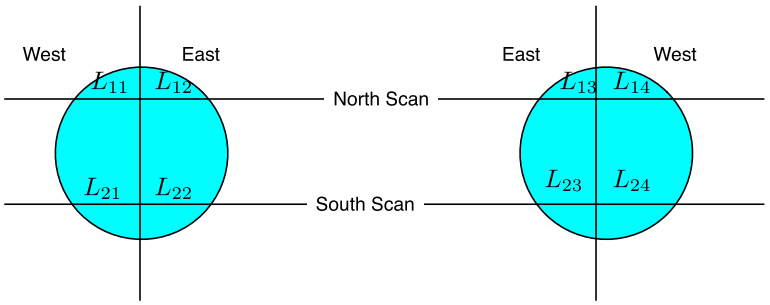


Figure 11.4 Nomenclature for measured chord parts.

The sensor also does some preprocessing of the roll and pitch angles. Depending on the number of edges sensed on each scan, roll, and pitch can be "latched" to the minimum or maximum output value or to zero to prevent erroneous output. A simple glitch filter also processes the measured values, which holds the output constant whenever there is a transition to or from the maximum or minimum value or a value latched to zero.

11.4.4 Analog Sun sensors

Analog Sun sensors are used in pairs to get attitude information. The sensor geometry is illustrated in Fig. 11.5.

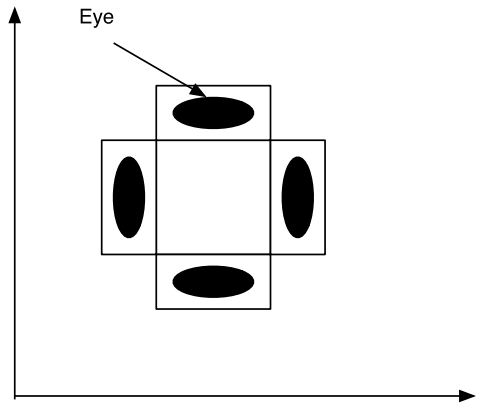


Figure 11.5 Sun-sensor geometry.

Two sensors' boresights are in the yz-array plane the other two are in the xz-array plane. The first is canted 45 deg from $+y$ and the other 45 deg from $+z$. The difference between the outputs of the sensors is an indication of the angle. Each sensor measures,

to first order, the cosine of the angle between the Sun vector and the sensing element. The measurement from one sensor is (in its simplest form)

$$y = g\cos\theta \tag{11.12}$$

$$\cos\theta = n^T u_{Sun} \tag{11.13}$$

where g is a gain. Generally, analog sensors are modeled as

$$y = \sum_{k=0}^{N} g_k \cos^k\theta \tag{11.14}$$

Since a single sensor is just a cosine detector it cannot give direction. If two such sensors are mounted together as shown in Fig. 11.6 the direction can be found.

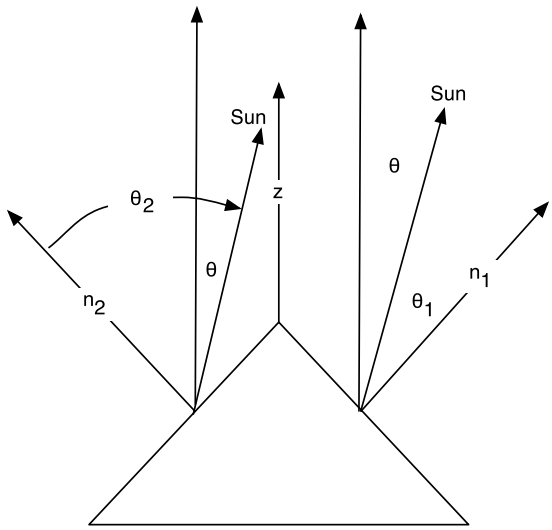


Figure 11.6 Simple one-dimensional Sun sensor.

$$\theta_1 = \cos^{-1}(n_1^T u_{Sun}) \tag{11.15}$$

$$\theta_2 = \cos^{-1}(n_2^T u_{Sun}) \tag{11.16}$$

If the angle between z and the sensor normal is β then

$$\theta_1 = \theta + \beta \tag{11.17}$$

$$\theta_2 = \beta - \theta \tag{11.18}$$

so that

$$\theta = \frac{\theta_2 - \theta_1}{2} \tag{11.19}$$

which gives sign information.

11.4.5 Digital Sun sensors

Digital Sun sensors measure the position of the Sun image in the sensor plane. These may use special patterns of thermal detectors in the sensor plane or may use CCD-type elements.

11.5. Gyros

Noise in a gyro or any physical system can be classified as:

1. White Gaussian;
2. Integrating white Gaussian produces random walk;
3. Integrating random walk produces random run;
4. By differentiating white noise we get phase noise;
5. Flicker noise is a Markov model

$$\dot{x} + \frac{1}{\tau}x = n \tag{11.20}$$

where τ is a time constant and n is Gaussian white noise;

6. By differentiating flicker noise we get flicker phase noise.

Allan Variance was initially created to study the frequency stability of oscillators. It is used for sensors to measure the time domain stability of a time signal due to noise. The Allan Deviation is the square root of the Allan Variance. Fig. 11.7 shows the Allan Deviation for the different types of noise.

The STIM300 is a MEMS gyro with noise parameters given in Table 11.2.

Table 11.2 STIM300 MEMS gyro noise.

Parameter	Value
Bias Run–Run	4 deg/h
Drift Rate Stability	3 deg/h
Bias Instability	0.3 deg/h
Angular Random Walk	0.15 deg/$\sqrt{\text{h}}$

Bias Run–Run is the variation of the bias between runs. This, along with the bias, is removed by calibration when the gyro is turned on. Angular random walk is white noise on the rate. This causes the angle, which is the integral of rate, to have a random-walk

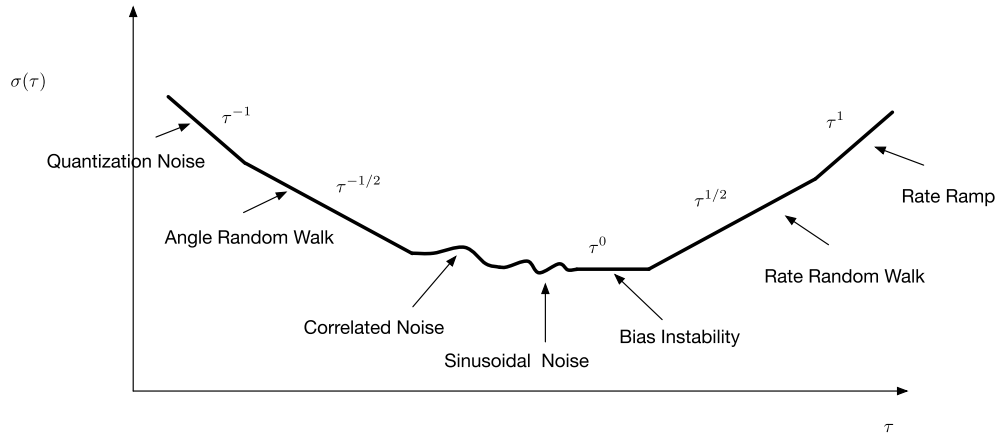


Figure 11.7 Allan Deviation and noise. τ is the averaging bin size.

element. The bias of a MEMS gyro wanders over time due to flicker in the electronics. Flicker noise is typically pink, that is it has a power spectral density that goes as the inverse of frequency. The power spectral density is the measure of a signal's power content versus frequency. The power spectral density of white noise is independent of frequency.

Table 11.3 summarizes the types of noise and how they match the MEMS gyro.

In Example 11.3 flicker pink noise is modeled by a first-order filter

$$y_k = y_{k-1} - \sigma y_{k-1} + \sigma u_k \tag{11.21}$$

where $\sigma = 0.1$ in this case.

The model that applies to the STIM300 single-degree-of-freedom gyro is

$$\dot{\theta} = \omega + b + f + \eta_\theta \tag{11.22}$$

$$\dot{f} = -\gamma f + \eta_f \tag{11.23}$$

$$\dot{b} = \eta_b \tag{11.24}$$

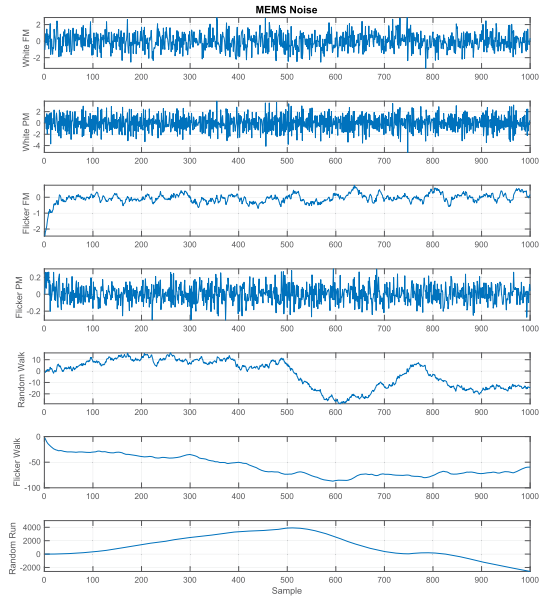
where θ is the integrated gyro angle, ω is the true inertial rate in body axes, b is the gyro bias, f is the bias instability, η_θ causes the integrated attitude to random walk, and η_b causes the bias to drift. The output of the gyro is both angular rate and, if it is a rate-integrating gyro, angular increments. The noise number for random walk, η_r is given in deg/$\sqrt{\text{h}}$. The conversion to a 1-sigma noise number is

$$n = \left(\frac{\pi}{180}\right)\left(\frac{\eta_r}{60}\right) \tag{11.25}$$

Table 11.3 Gyro noise with frequency and time bin size dependence. "P" means phase and "F" frequency.

Noise	Spectrum	Allan Deviation	Alan Variance	Modified Allan Deviation	Modified Allan Variation	Time Deviation	Time Variation	MEMS
White PM	f^2	τ^{-1}	τ^{-2}	$\tau^{-3/2}$	τ^{-3}	$\tau^{1/2}$	τ^{-1}	
Flicker PM	f	τ^{-1}	τ^{-2}	τ^{-1}	τ^{-2}	τ^{0}	τ^{0}	
White FM	f^0	$\tau^{-1/2}$	τ^{-1}	$\tau^{-1/2}$	τ^{-1}	$\tau^{1/2}$	τ^{1}	
Flicker FM	f^{-1}	τ^{0}	τ^{0}	τ^{0}	τ^{0}	τ^{1}	τ^{2}	Bias Instability
Random Walk	f^{-2}	$\tau^{1/2}$	τ^{1}	$\tau^{1/2}$	τ^{1}	$\tau^{3/2}$	τ^{3}	Angular Random Walk
Flicker Walk	f^{-3}	τ^{1}	τ^{2}	τ^{1}	τ^{2}			Drift-Rate Stability
Random Run	f^{-4}	$\tau^{3/2}$	τ^{3}	$\tau^{3/2}$	τ^{3}			

```
%% MEMS noise

yL = {'White FM' 'White PM' 'Flicker FM'...
      'Flicker PM' 'Random Walk'...
      'Flicker Walk' 'Random Run'};

n  = 1000;
nS = zeros(7,n);

sF = 0.1;

nS(1,:) = randn(1,n);
nS(2,:) = [0,diff(nS(1,:))];
nS(3,:) = Filter(sF,randn(1,n));
nS(4,:) = [0,diff(nS(3,:))];
nS(5,:) = cumsum(nS(1,:) );
nS(6,:) = cumsum(nS(3,:) );
nS(7,:) = cumsum(nS(5,:) );

Plot2D(1:n,nS,'Sample',yL,'MEMS Noise')

% A filter to simulate pink noise
function y = Filter(sig,u)

n = length(u);
y = zeros(1,n);
y(1) = u(1);

for k = 2:n
  y(k) = y(k-1) - sig*y(k-1) + sig*u(k);
end

end
```



Example 11.3: Noise sources.

Angular increments can be converted to rates using one of the following formulas [1],

$$\omega_k = \frac{a_k}{\Delta T} \tag{11.26}$$

$$\omega_k = \frac{3a_k - a_{k-1}}{\Delta T} \tag{11.27}$$

$$\omega_k = \frac{11a_k - 7a_{k-1} + 2a_{k-2}}{\Delta T} \tag{11.28}$$

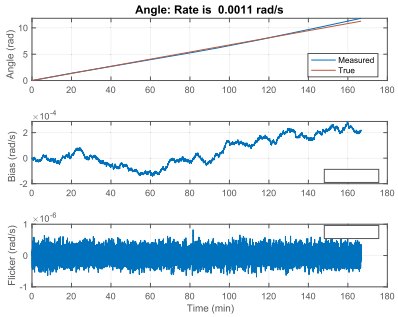
where a_k is the angular increment at step k, that is

$$a_k = \theta_k - \theta_{k-1} \tag{11.29}$$

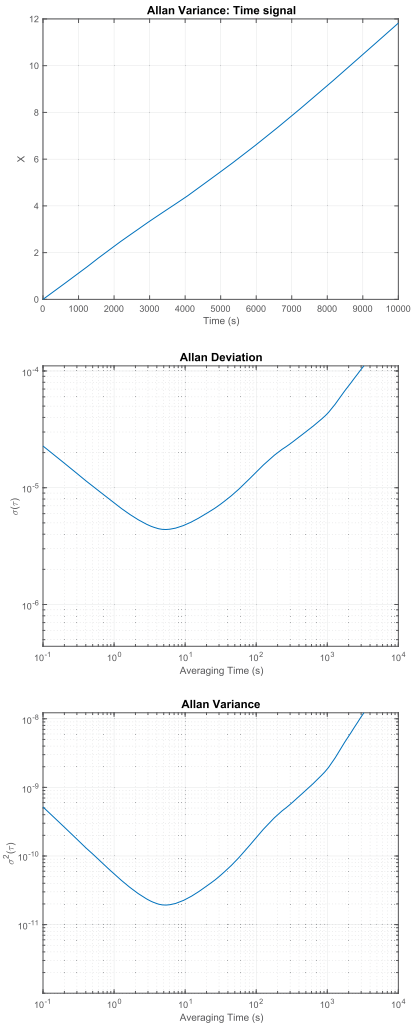
and ΔT is the time step or the time interval between step k and step $k-1$.

Example 11.4 shows simulation results, Allan Variance, and Allan Deviation for the STIM300 model. The drift-rate stability, bias instability, and angular random walk are clearly shown in the plots.

Note that the MEMS gyro has a large bias. This must be measured when the gyro is turned on and stored as a constant. This should be done before the release of the spacecraft from the ISS.




```
%% Simulate a MEMS gyro
el                  = ISSOrbit;
degPHrToRadPs       = pi/180/3600;
n                   = 1e5;          % Number of steps
dT                  = 0.1;          % sec
d                   = RHSMEMSGyro;
d.rate              = 2*pi/Period(el(1));   % rad/s
d.angleNoise        = RW2SDev(0.15);
d.flickerNoise      = 0.3*degPHrToRadPs;
d.biasNoise         = 3*degPHrToRadPs;


%% Run the simulation
xP                  = zeros(3,n);
x                   = zeros(3,1);

for k = 1:n
  xP(:,k)           = x;
  x         = RK4(@RHSMEMSGyro,x,dT,0,d);
end

t       = (0:n-1)*dT;
[tS,tL] = TimeLabl(t);

yL = {'Angle (rad)', 'Bias (rad/s)' 'Flicker (rad/s)'};

s  = sprintf('Angle: Rate is %7.4f rad/s',d.rate);
lL = {'Measured', 'True'};
Plot2D(tS,[xP;d.rate*t],tL,yL,s,'lin',{'[1 4]' '[2]' '[3]'},[],[],[],[],lL,{''},{''});

%% Allan variance
AllanVariance(xP(1,:),t);
Figui
```



Example 11.4: STIM300.

11.6. Other sensors

11.6.1 Magnetometers

Magnetometers are used for attitude determination and for measuring the magnetic field for adjusting the gain of magnetic-torquer systems. Magnetometers measure the total magnetic field and must be kept away from sources of magnetic fields. Usually, they are mounted on booms.

The magnetic-torque rods will produce magnetic fields that will be seen by the magnetometer. Example 11.5 shows the effect on the measured magnetic field. Two cases are shown. One has the magnetometer at the opposite corner of the bus. The second puts it on a 25-cm boom along the -Y direction. Putting the magnetometer on the boom reduces the torquer field by half. Given that the field is deterministic, we can measure it during integration and test and subtract it whenever a torquer is turned on. The magnetic field due to a dipole is

$$b = \frac{\mu_0}{4\pi}\left(\frac{3r(M^T r)}{|r|^5} - \frac{M}{|r|^3}\right) \tag{11.30}$$

This formula assumes an infinitesimal dipole so it is only applicable when the point is a distance from the current loop. The field drops off as $|r|^3$ so its effect can be mitigated by distance. For this reason, magnetometers are often placed on booms. Well-known formulas exist for the magnetic field for a finite solenoid, [2], that would apply to a torquer bar.

Example 11.5 shows the effect of a magnetic dipole on the measured magnetic field in an ISS orbit. The first plot shows the field around a dipole. The second shows the measured magnetic field. The field due to the torquer can be measured and subtracted from the measurement. A bandpass filter, with the center frequency at orbit rate, can also be used to reduce errors.

There will be other sources of magnetic fields, such as current loops due to the spacecraft power bus and the motors in the reaction wheels.

11.6.2 Accelerometers

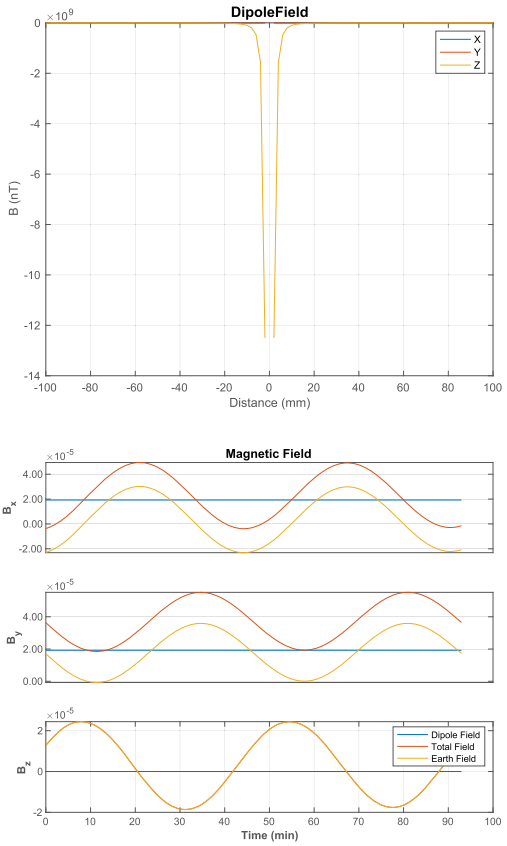
Accelerometers measure the acceleration of a mass relative to their base. A simple proof-mass accelerometer is shown in Fig. 11.8.

The mass is attached to the base via a spring and damper. The transfer function is

$$\frac{s^2}{s^2 + 2\zeta\omega s + \omega^2} \tag{11.31}$$

At low frequencies, the accelerometer has no response. At high frequencies, it has a transfer function of unity.

```
%% Local magnetic field

r = [linspace(-0.1,0.1,101);zeros(2,101)];

b = MagneticFieldDipole([0;0;1],r);

Plot2D(r(1,:)*1000,b*1e9,'Distance (mm)','B (nT)'
    ,'DipoleField')
legend('X', 'Y','Z')

% All three torquers are on
dipole     = 1; % ATM^2

% Unit vectors for the dipoles
u          = eye(3);

% Points on corner of spacecraft and on a boom
p          = [10;10;10]*0.01;

% ISS Orbit
[el,jD0]   = ISSOrbit;
[r,~,t]    = RVOrbGen(el);

% Magnetic field
jD         = jD0 + t/86400;
bEarth     = BDipole(r,jD);
b          = MagneticFieldDipole([0;0;1],p);

bT         = bEarth + b;
yL = {'B_x' 'B_y' 'B_z'};
n          = length(jD);

TimeHistory(t,[b*ones(1,n);bT;bEarth],yL,'
    Magnetic Field',{[1 4 7] [2 5 8] [3 6 9]})
    ;
legend('Dipole Field', 'Total Field', 'Earth
    Field')
```



Example 11.5: Effect of the magnetic-torque rods on the measured magnetic field.

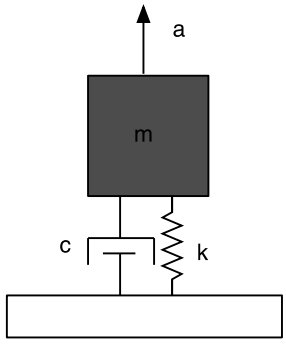


Figure 11.8 Proof-mass accelerometer.

11.6.3 Potentiometers

Potentiometers measure angles by changes in resistance. They usually are low-accuracy sensors and may have a large deadband around zero. Many produce a signal when the device passes through zero, which is more accurate than the angle measurements. Usually, potentiometers are used as a backup for other measurements, such as counting steps from a stepping motor.

11.6.4 Angle encoders

Angle encoders are either absolute or relative. Absolute angle encoders give you the actual measurement from a fixed reference. Relative give you the change in angle. Optical angle encoders are often made from a pattern painted on a glass disk. Encoders are expensive but are necessary for some high-precision applications, such as sinusoidal motor control.

11.7. Star cameras

11.7.1 Pinhole camera

The lowest-order approximation to the optical system is a pinhole camera. The pinhole camera is shown in Fig. 11.9.

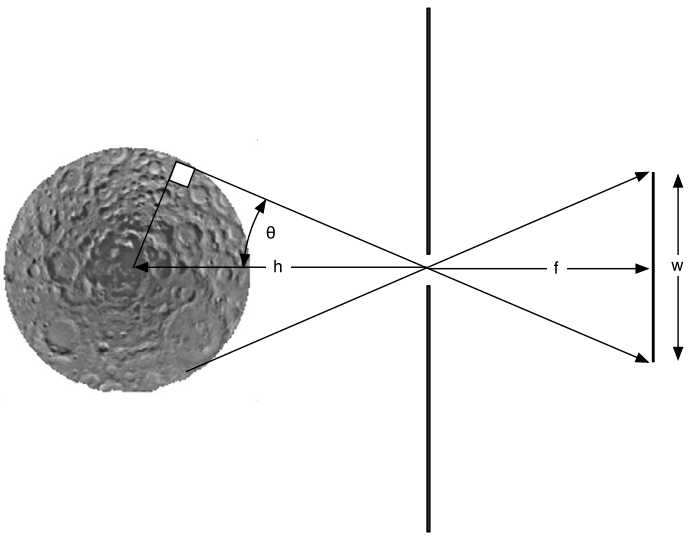


Figure 11.9 Pinhole camera.

A point $P(X, Y, Z)$ is mapped to the imaging plane by the relationships

$$u = \frac{fX}{Z} \tag{11.32}$$

$$v = \frac{fY}{Z} \tag{11.33}$$

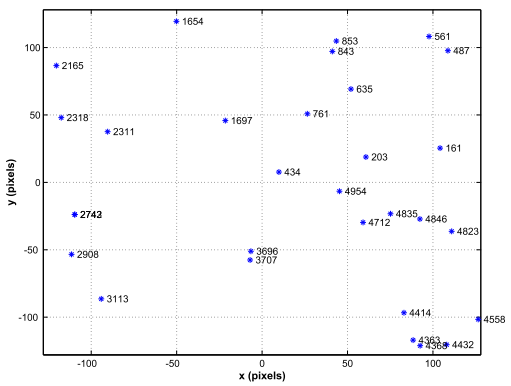
where u and v are coordinates in the focal plane, f is the focal length, and Z is the distance from the pinhole to the point along the axis normal to the focal plane. This assumes that the Z-axis of the coordinate frame X, Y, Z is aligned with the boresight of the camera.

The angle that is seen by the imaging chip is

$$\theta = \tan^{-1}\left(\frac{w}{2f}\right) \tag{11.34}$$

where f is the focal length. The shorter the focal length, the larger the image. The pinhole camera does not have any depth of field, but that is unimportant for many far-field imaging problems. The field-of-view of a pinhole camera is limited only by the sensing element. An example of a pinhole image from the Hipparcos catalog (showing only stars with a visual magnitude < 6) is given in Example 11.6.

```
1 PinholeCamera
2 PrintFig(1,1,1,'PinholeCameraDemo');
```



Example 11.6: Pinhole-camera star image.

Errors associated with the star camera are shown in Fig. 11.10. These errors are discussed in the following sections.

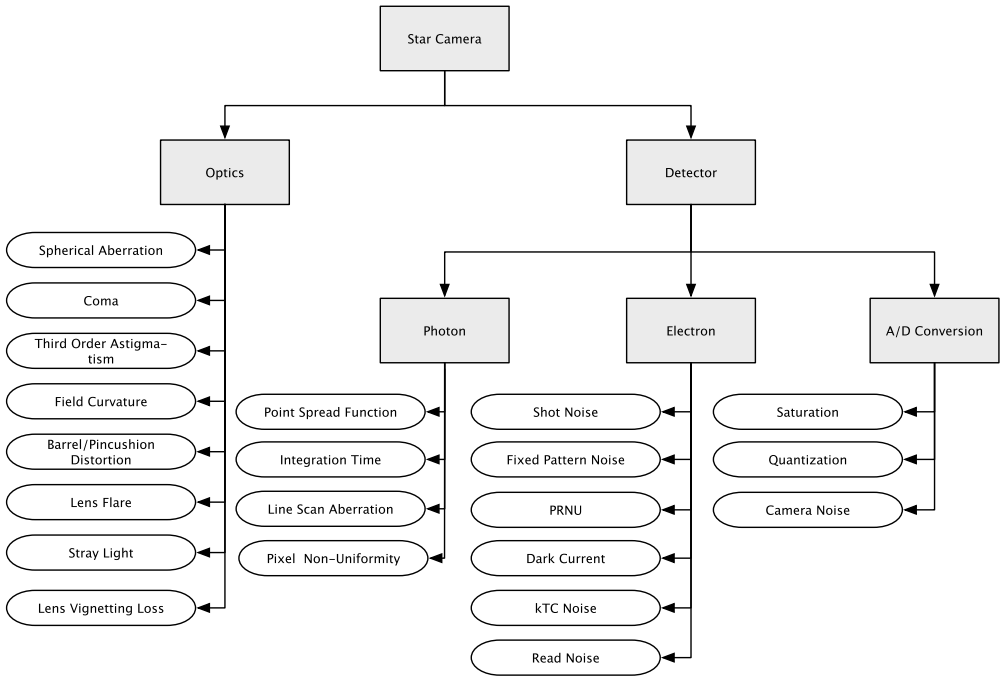


Figure 11.10 Star-camera errors.

11.7.2 Optical errors

Optical errors in a camera are due to the telescope barrel and lens. Optical errors are given in Table 11.4.

Table 11.4 Optical-error budget terms.

Error Source	Bias	Long	Orbit Rate	Short	Description
Spherical aberration	×			×	This is the difference in focal length between axial rays and rays along the periphery of aperture.
Coma	×			×	This is a geometric property in which offaxis point sources do not all come together at the same point.
Third-order astigmatism	×			×	This is a geometric effect even in symmetric optical systems.
Field curvature	×			×	The image really lies on the surface of a sphere, not a flat plate. This causes offaxis errors when an imaging chip is used.

continued on next page

Table 11.4 (*continued*)

Error Source	Bias	Long	Orbit Rate	Short	Description
Barrel/Pin-cushion distortion	×			×	Magnification decreases with distance from the optical axis.
Lens flare				×	Light scattered in a telescope by internal reflection in lens.
Stray light				×	Scattering when a bright source, such as the Sun, is in the field of view. Diffraction features are caused by support structures, baffles, and other objects.

11.7.3 Imaging-chip errors

Imaging-chip errors are given in Table 11.5 [3]. These errors can be mitigated by

1. Varying the pixel integration time;
2. Combining frames;
3. Calibrating.

Table 11.5 Imaging-chip error budget terms.

Error Source	Bias	Long	Cyclic	Short	Description
Point Spread Function				×	Errors in the point spread function will reduce the fitting accuracy.
Integration Time				×	Long integration times can cause overexposure resulting in saturated pixels.
Line-Scan Aberration				×	A time delay from when the top and bottom rows are integrated due to rolling shutters.
Pixel Nonuniformity	×				This will cause a bias since the actual light measured is only over a portion of the pixel.
Shot Noise				×	Shot noise is caused by the random arrival of photons. This is a fundamental trait of light. Since each photon is an independent event, the arrival of any given photon cannot be precisely predicted; instead the probability of its arrival in a given time period is governed by a Poisson distribution.
Fixed-Pattern Noise				×	The is temporally constant pixel nonuniformity in noise.

continued on next page

Table 11.5 (*continued*)

Error Source	Bias	Long	Cyclic	Short	Description
PRNU				×	Photo Response Nonuniformity cause errors since each pixel does not produce the same response for the same number of photons.
Dark Current				×	This is current that flows in the pixel in the absence of light. It increases with time.
kTC Noise				×	Thermal noise.
Read Noise				×	These are onchip sources of noise that are combined into the Gaussian noise generated when a pixel is read.
Saturation				×	If a pixel saturates it will cause an error in the centroiding fit. This is a problem if multiple pixels in a centroid are saturated.
Quantization				×	Limits the accuracy of the centroiding computation due to minimum bit value.
Camera noise				×	All sources of noise that come between the readout of the pixel and the A/D conversion.

PRNU and fixed-pattern noise are characterized as random noise but they remain constant for each image once the detector is turned on.

11.8. GPS

GPS receivers can use any one of several navigation-satellite constellations. Four are

1. Global Positioning Systems (US);
2. GLONASS (Russia);
3. BeiiDou (PRC);
4. Galileo (ESA).

Many receivers use all of them. Most GPS receivers provide a navigation solution but often allow access to the raw data. For attitude determination, raw signals are required. One or more antennas are required. When multiple antennas are used, algorithms use GPS differential carrier-phase measurements between the antennas to determine the attitude [4]. Antenna geometry is an important factor in accuracy.

References

[1] E. Wong, W. Breckenridge, Inertial attitude determination for dual-spin planetary spacecraft, Journal of Guidance 6 (6) (1983) 491–498.

[2] E.E. Callaghan, S.H. Maslen, The magnetic field of a finite solenoid, https://www.osti.gov/biblio/4121210, 10 1960.

[3] M. Knutson, D. Miller, Fast Star Tracker Centroid Algorithm for High Performance CubeSat with Air Bearing Validation, Master's thesis, M.I.T. Department of Aeronautics and Astronautics, June 2013, SSL # 5-12.
[4] Charles Wang, Rodney A. Walker, Werner Enderle, Single antenna attitude determination for FedSat, in: Proceedings Institute of Navigation GPS-02, 2002, pp. 1–12.

CHAPTER 12

Attitude control

12.1. Space story

A new type of thruster was the Improved Electrothermal Hydrazine Thruster (IMPEHT). In the previous spacecraft, monopropellant thrusters were used for attitude control during North/South station keeping and Electrothermal Hydrazine Thruster (EHTs) were used to produce the velocity change. IMPEHTs could be pulsed and were more efficient so they could be used for both, with the bonus of higher efficiency. When the first maneuver was done the attitude was jittery. After much analysis, this was determined to be due to helium bubbles from the blowdown-system pressurant mixed with the fuel. The more efficient IMPEHTs had smaller feed tubes so that some pulses were just hot helium. Attempts to "burp" the spacecraft proved unsuccessful. IMPEHTs were replaced by arc jets on subsequent spacecraft.

12.2. Introduction

This chapter discusses topics of control particularly relevant to spacecraft attitude control. Attitude control is a vast subject. In its most general form, it encompasses controlling the orientation of the spacecraft relative to a target orientation and the orientation of parts of the spacecraft relative to a target orientation or to the spacecraft itself. The target may be fixed in the inertial frame or fixed to a planet fixed frame. It may also be moving relative to either of those frames. When we talk about 3-axis control it means we want to control all three axes of the spacecraft with respect to the target orientation. Single-axis control (which is the case with a spinner) only requires control of one axis with respect to a target orientation. Fig. 12.1 shows a hierarchy of typical attitude control systems.

Attitude control systems can be passive or active. Active means that we take a measurement and compute a control action that is accomplished with an actuator. The computation can be analog or digital, although most modern spacecraft use digital control. A passive system uses the physics of the spacecraft and possibly a control actuator to perform control. For example, the dual-spin-turn involves turning on a momentum wheel mounted transversely to the axis of rotation. The momentum of the spacecraft is transferred to the wheel and the spacecraft nutates and then flips on its side. The nutation is usually damped through fuel slosh. This maneuver is entirely passive yet achieves the desired pointing. A gravity-gradient-stabilized spacecraft is another example of passive control. A libration damper damps excursions from the nominal attitude.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00023-1

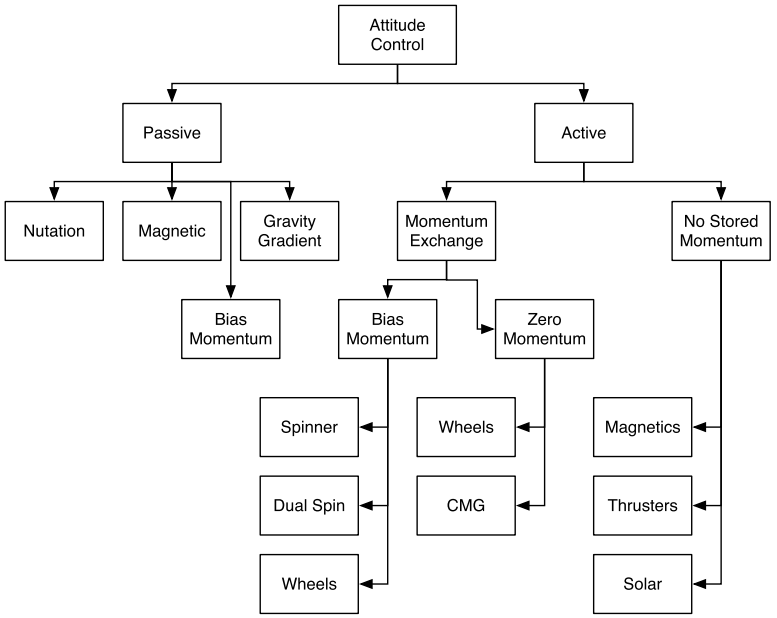


Figure 12.1 Attitude control system hierarchy. This diagram shows various types of attitude control systems.

The spacecraft dynamics, including gravity gradient, have oscillatory modes about pitch and roll.

Active control systems either can transfer momentum within the spacecraft or use external torques produced by thrusters, solar vanes, or magnetic torquers, to control the spacecraft. Momentum exchange is easy to understand. If the spacecraft is inertially fixed and you spin up a wheel the momentum going into the wheel comes from the spacecraft that rotates in the opposite direction to conserve momentum. This is the basis of the reaction wheel. No momentum is exchanged with the environment. Only energy is consumed. This is a very convenient method for reorienting a spacecraft or controlling the orientation in the face of cyclic disturbances. The momentum-less type reacts against the environment to produce torque. Magnetic torquers only use electricity and solar vanes use power to orient the vanes to produce the desired torque. Thrusters consume fuel. Momentum-exchange systems also require actuators to control inertially fixed torques that change the spacecraft momentum. These can be any of the actuators required by the momentum-less systems.

The rest of this chapter discusses attitude control systems in general and then covers some interesting examples of control systems.

12.3. Attitude control phases

Fig. 12.2 shows typical phases in the life of a spacecraft. The spacecraft starts on top of whatever vehicle delivers it to its initial orbit. The transfer vehicle may be 3-axis stabilized or spinning. If it is spinning the spacecraft needs to despin (if it is not a spinner) and remove any other angular rates. While it does this, it needs to maintain an orientation where there is Sun on the solar panels.

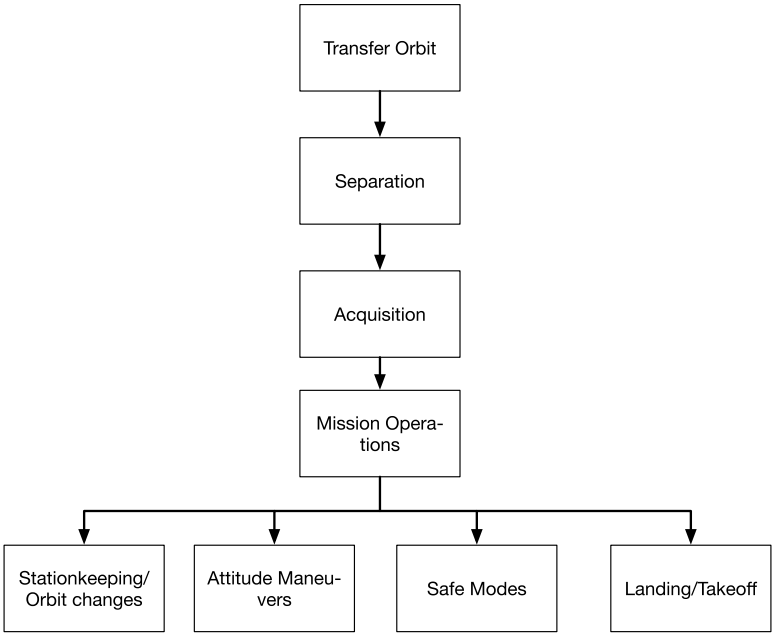


Figure 12.2 Attitude control phases.

After despin and detumbling the spacecraft needs to acquire its baseline attitude. This may require some kind of large-angle maneuver. Once it is in its nominal orientation the mission begins. For some spacecraft, the launch vehicle does not leave it in its final orbit and a series of propulsive maneuvers, lumped in with station keeping on the graph, must be done. This will require keeping the delta-V thrusters in a specific orientation. Station keeping may be required during the mission life. This is always necessary for geosynchronous spacecraft that must stay in their geosynchronous "box".

Some spacecraft, for example, the NASA Hubble or James Webb Space Telescopes, must reorient to look at different targets. Others, like communications satellites, just point at the Earth. Some Earth-resources satellites need to orient sensors at particular points on the ground. This may involve large and rapid attitude maneuvers.

The final phase is known as a safe mode. Safe means there is power on the spacecraft. This leads to the term "Sun-safe mode" where the spacecraft spins around the Sun line with its solar panels pointed at the Sun. Not all spacecraft have safe modes.

12.4. Attitude control system

Fig. 12.3 shows a hierarchy of typical attitude control systems. In some systems, the blocks represent real blocks of software or analog hardware. In most systems, the block functions are distributed among many different pieces of software. The first step is to get information from the sensors. In some cases, this data can be used directly. For example, an Earth sensor will produce roll and yaw angles that can be used directly in the control algorithms, albeit with some filtering. In other cases, such as with a star tracker or camera, the positions of the stars on the focal plane must be converted into an attitude representation through further processing.

Some systems employ an estimator that is simply a filter that employs a model of the system. The model may be only of the sensor hardware or may be one of the spacecraft. The output is used to produce desired control accelerations. These must be converted to torque demand. For example, if the inputs are roll, pitch, and yaw and the outputs are angular accelerations, the desired torque can be computed from

$$T = Ia \tag{12.1}$$

where a is the control acceleration vector and I is the inertia matrix. If there are off-diagonal terms the torque will include contributions from one or more axis acceleration. The torque demand must be converted into actuator commands. In the case of reaction wheels, the 3-axis torque must be converted to 3 or more reaction-wheel torque commands. If thrusters are used these commands might be used to produce pulsewidth demands for 3 or more thrusters. This is done by the torque-distribution block. Finally, the actuator demands must be converted to a form suitable for use by the hardware in the actuator interface block.

A control system must always accommodate ground commands. These may be commands that go directly to hardware, in some cases bypassing the flight computer. In other cases, they may be additional axis-acceleration demands.

12.5. Single-axis control

It is instructive to design a controller for a single axis of a spacecraft. In many cases, such as in station keeping, this is common practice. As noted above, if our controller works with angular accelerations we can ignore cross-axis coupling. The single-axis controller is shown in Fig. 12.4. The control system is in series with the spacecraft dynamical

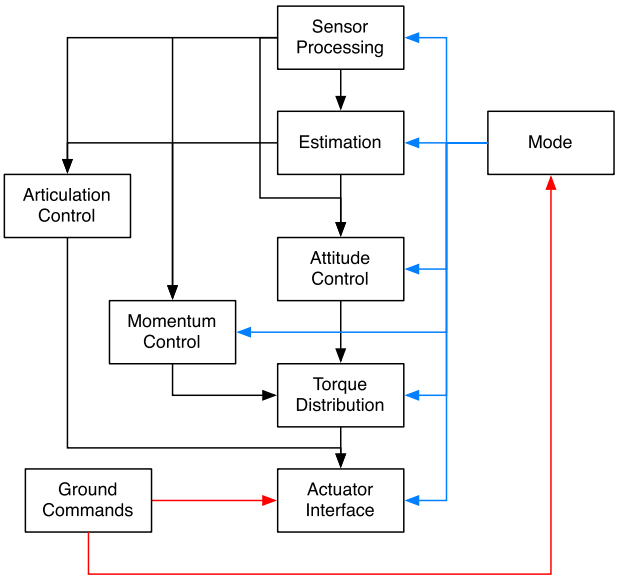


Figure 12.3 Attitude control system architecture.

model. We take measurements and feed them back, with a negative sign, to correct errors.

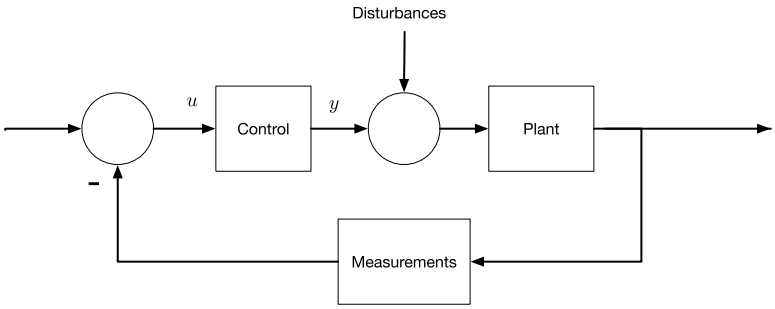


Figure 12.4 Control system.

If we model our system as a double integrator we get

$$\ddot{\theta} = a(t) \tag{12.2}$$

where θ is the angle and a is the angular acceleration. The Laplace transform is

$$s^2\theta = a(s) \tag{12.3}$$

where $s = j\omega$. $j = \sqrt{-1}$. With this model a constant acceleration will cause the angle to grow without bound. During a station-keeping maneuver this happens because the delta-V thrusters produce a constant disturbance acceleration.

Three possible controllers are, assuming we only measure angle (γ is an angle in this case)

$$\frac{\gamma}{u} = k \tag{12.4}$$

$$\frac{\gamma}{u} = k\left(1 + \frac{\tau_r s}{1 + \tau_f s}\right) \tag{12.5}$$

$$\frac{\gamma}{u} = k\left(1 + \frac{\tau_r s}{1 + \tau_f s} + \frac{1}{\tau_I s}\right) \tag{12.6}$$

The first is a proportional controller with a forward gain of k. The second adds a derivative, that is filtered. The derivative time constant is τ_r and the filter time constant is τ_f. Filtering the derivative term is done in industrial controllers because differentiation amplifies noise so you want to cut off the differentiation at some frequency. Another way of looking at it is that there is nothing you want to control at higher frequencies, thus everything about it is the noise. The third adds an integrator with a time constant of τ_I. This time constant is roughly how long the controller will take to integrate out the constant disturbance. Bode shows the Bode plots of the open-loop controllers in series with the open-loop system. The pure gain does not change the phase but just shifts the magnitude up. The derivative adds phase shift at the crossover frequency, where the magnitude is one. This shows that it is adding damping. The integral shows the gain increasing as s approaches zero, demonstrating integral action. Example 12.1 shows Bode and time–response plots.

With the first controller, the angle excursions will be limited but the spacecraft will oscillate. The second will damp the motion but there will be an offset. The third will eventually drive the error to zero.

It is instructive to look at how the open-loop response relates to the closed-loop response. The error that is the input to the control system is

$$e = u - \gamma \tag{12.7}$$

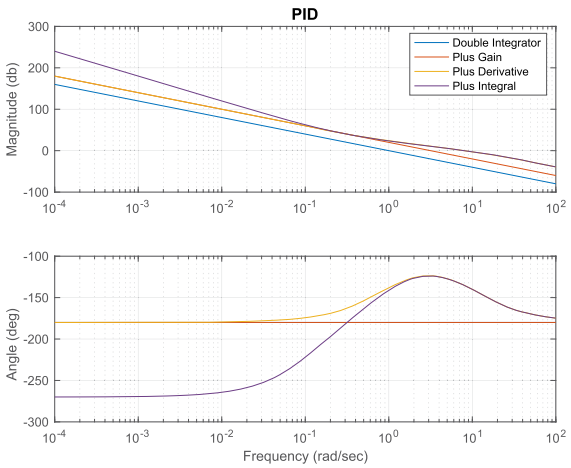
and

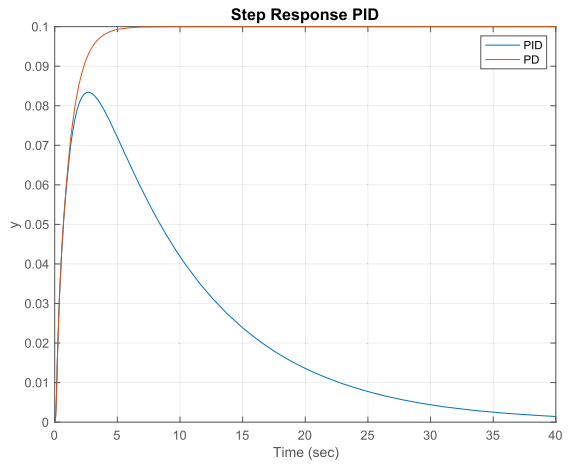
$$\gamma = CPe \tag{12.8}$$

Solving for γ

$$\frac{\gamma}{u} = \frac{C}{\frac{1}{P} + C} \tag{12.9}$$

```
%% PID Control


w = logspace(-4,2);


% Double integrator
aS              = [0 1;0 0];
bS              = [0;1];
cS              = [1 0];
dS              = 0;
[m,p]           = FResp(aS,bS,cS,dS,1,1,w);
m               = 20*log10(m);

% Add gain
aP              = 0;
bP              = 0;
cP              = 0;
dP              = 10;
[a, b, c, d]    = Series(aP,bP,cP,dP,aS,bS,cS,dS);
[m2,p2]         = FResp(a,b,c,d,1,1,w);
m2              = 20*log10(m2);

% Add derivative
tauF            = 0.1;
k               = 10;
tauR            = 1;
[aR,bR,cR,dR]   = ND2SS(k*[tauR+tauF 1],[tauF 1]);
[a, b, c, d]    = Series(aR,bR,cR,dR,aS,bS,cS,dS);
[m3,p3]         = FResp(a,b,c,d,1,1,w);
m3              = 20*log10(m3);

% Add Integrator
tauI            = 10;
f               = k/tauI;
[aP,bP,cP,dP]   = ND2SS(f*[tauI*(tauR+tauF) tauI+
      tauF 1],[tauF 1 0]);
[a, b, c, d]    = Series(aP,bP,cP,dP,aS,bS,cS,dS);
[m4,p4]         = FResp(a,b,c,d,1,1,w,'unwrap');
m4              = 20*log10(m4);


%% Plot
yL    = {'Magnitude (db)';'Angle (deg)'};

Plot2D(w,[m;m2;m3;m4;p;p2;p3;p4],'Frequency (rad/
      sec)',yL,'PID',['xlog';'xlog'],{'[1 2 3 4]
      ' '[5 6 7 8]'});

legend('Double Integrator','Plus Gain', 'Plus
      Derivative', 'Plus Integral')



n    = 2000;
dT   = 0.02;
y    = zeros(1,n);
u    = 1;

[aC, bC, cC, dC] = CLoopS( aS,bS,cS,aP,bP,cP,dP )
      ;
[aC,bC] = C2DZOH(aC,bC,dT);

x = zeros(length(aC),1);
yI = zeros(1,n);
for k = 2:n
  yI(1,k) = cC*x + dC*u;
  x       = aC*x + bC*u;
end

[aC, bC, cC, dC] = CLoopS( aS,bS,cS,aR,bR,cR,dR )
      ;
[aC,bC] = C2DZOH(aC,bC,dT);

x = zeros(length(aC),1);
yR = zeros(1,n);
for k = 2:n
  yR(1,k) = cC*x + dC*u;
  x       = aC*x + bC*u;
end

[t,tL] = TimeLabl((0:n-1)*dT);

Plot2D(t,[yI;yR],tL,'y','Step Response PID');
legend('PID','PD')
```





Example 12.1: Bode and time–response plots.

If $C = 1$ given the double-integrator plant the resulting closed-loop system is

$$\frac{y}{u} = \frac{1}{s^2 + 1} \tag{12.10}$$

where $s = 0$; it is no longer ∞. The denominator is that for an undamped oscillator. This is what the open-loop crossover frequency shows. The phase shift was 180 degrees at the crossover.

If $C = k(\zeta\sigma s + \sigma^2)$ we get

$$\frac{y}{u} = k\frac{\zeta\sigma s + \sigma^2}{s^2 + k\zeta\sigma s + k\sigma^2} \tag{12.11}$$

which is a damped oscillator.

12.6. Three-axis control

Each axis is controlled independently with a proportional-integral-differential controller. The basic system is shown in Fig. 12.5. The reference quaternion and measured quaternion are multiplied to produce small angles. The proportional-integral-differential controller produces a per-axis acceleration demand. This is multiplied by the inertia matrix to produce a torque demand. If we are using thrusters we convert this to a pulsewidth demand using linear programming, also known as the simplex algorithm. If we are using reaction wheels we use the pseudoinverse to convert torque demand to reaction-wheel torque demand. This is further converted into a voltage.

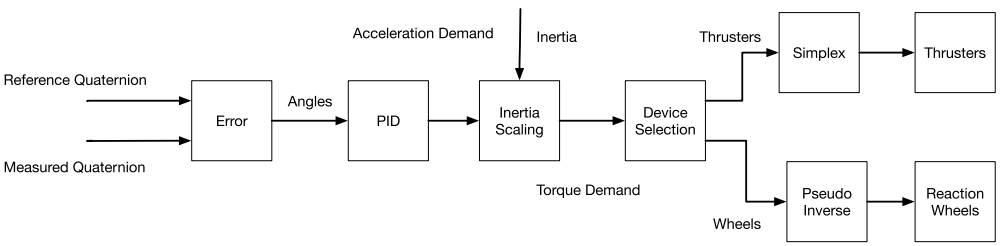


Figure 12.5 Attitude control system flow.

The rigid-body equations are

$$T = I\dot{\omega} + \omega^{\times} I\omega \tag{12.12}$$

If rates are small and the angles are small this reduces to

$$T = I\ddot{\theta} \tag{12.13}$$

If we define

$$T = Iu \tag{12.14}$$

then the axes are decoupled and the control problem is reduced to

$$u = \ddot{\theta} \tag{12.15}$$

or three decoupled double integrators. This is valid if the spacecraft angular rates are small and other modes, such as slosh or flexibility, are at frequencies much higher than the controller bandwidth. Once the control is computed the control torques are formed by the operation

$$T = Iu \tag{12.16}$$

which is a 3-by-3 matrix times a 3-by-1 vector multiplication. We can now design each axis control loop independently.

If we have two quaternions q_I and q_F that are the initial and final quaternions and we want to go from the first to the second we write

$$\Delta q = q_T^T \otimes q_I \tag{12.17}$$

As noted in the chapter on kinematics a quaternion is a unit vector and an angle about that unit vector. Therefore

$$\Delta q = \begin{bmatrix} \cos\frac{\phi}{2} \\ a_1 \sin\frac{\phi}{2} \\ a_2 \sin\frac{\phi}{2} \\ a_3 \sin\frac{\phi}{2} \end{bmatrix} \tag{12.18}$$

where a is the unit vector.

As an example, let the initial quaternion be

$$q_I = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12.19}$$

and the target is

$$q_T = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ 0 \end{bmatrix} \tag{12.20}$$

We want to do a single-axis rotation. We do not need quaternions for this. The transpose of q_I is q_I and the product of q_I and q_T is q_T. Our unit vector is then

$$a = \frac{\begin{bmatrix} -0.6 \\ 0 \\ 0 \end{bmatrix}}{0.6} \tag{12.21}$$

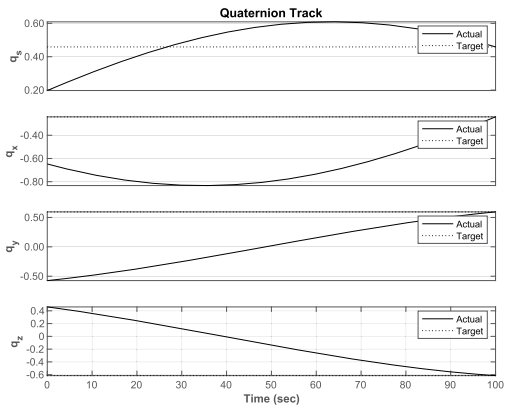
or a vector aligned with the x-axis.

This works for arbitrary initial and target quaternions. Example 12.2 shows how it works for randomly selected quaternions. The magnitude of the angular rate is

$$\omega = \frac{2\cos^{-1} q(1)}{t} \tag{12.22}$$

where t is the desired maneuver duration.

```
qI       = QRand;
qT       = QRand;

deltaQ   = QMult(QPose(qT),qI);
u        = Unit(deltaQ(2:4));
angle    = 2*acos(deltaQ(1));
t        = linspace(0,100);
n        = length(t);
omega    = u*angle/t(n);
dT       = t(2);
q        = zeros(4,n);
q(:,1)   = qI;

for k = 2:n
  q(:,k) = RK4(@QDot,q(:,k-1),dT,t(k),omega);
end

qL = {'q_s' 'q_x' 'q_y' 'q_z'};

lL = {'Actual' 'Target'};
TimeHistory(t,[q;qT.*ones(1,n)], qL, 'Quaternion␣
      Track',...
  {[1 5] [2 6] [3 7] [4 8]},{lL lL lL lL},1);


%% Right hand side;
function qDot = QDot( q, ~, omega )
qDot = QIToBDot(q,omega);
end
```



Example 12.2: Single-axis quaternion maneuver. The red lines are the target quaternion. The spacecraft reaches the target but does not stop at the target.

This is the most direct maneuver. Your spacecraft might have maneuver constraints, for example not having a star sensor to look at the Sun, in which case you would need to compute a less direct route.

This maneuver can be implemented in several ways. One is to break the maneuver into several chunks and use the PID to attain each delta angle. This makes your PID controller completely general-purpose as it is only dealing with the linear problem. An

alternative is to compute an angular rate, based on the momentum limits or torque limits of your control system. You would then track the desired angular rate and angle. Since the maneuver is single-axis you could also use a bang-zero-bang controller. That is, accelerate to the desired maneuver rate, coast, and then decelerate.

12.7. Gravity-gradient control

For many spacecraft, gravity-gradient control is sufficient to maintain Earth pointing. Torque is produced by the variation in gravity across the spacecraft. If the mass properties are correct, it will see restoring torques that will keep it nominally Earth pointing. Section 6.7 discusses gravity-gradient dynamics. As explained, a gravity-gradient-stabilized spacecraft has stiffness about all three axes, but no damping. Roll and yaw are coupled while pitch is decoupled. Thus we need roll/yaw and pitch damping. The dynamical equations are

$$T_x = I_x\ddot{\theta}_x + n\dot{\theta}_z(I_y - I_z - I_x) - 4n^2\theta_x(I_z - I_y) \tag{12.23}$$

$$T_y = I_y\ddot{\theta}_y + 3n^2(I_x - I_z)\theta_y \tag{12.24}$$

$$T_z = I_z\ddot{\theta}_z + n\dot{\theta}_x(I_z + I_x - I_y) - n^2\theta_z(I_y - I_x) \tag{12.25}$$

Since we only want to damp angular rates, a rate gyro is a suitable sensor. The control torques are

$$T_x = -4\zeta n\sqrt{\frac{(I_y - I_z)}{I_x}}\dot{\theta}_x \tag{12.26}$$

$$T_y = -2\zeta n\sqrt{\frac{3(I_x - I_z)}{I_y}}\dot{\theta}_y \tag{12.27}$$

$$T_y = -2\zeta n\sqrt{\frac{(I_x - I_y)}{I_z}}\dot{\theta}_z \tag{12.28}$$

where ζ is the desired damping ratio. We can study this simply be forming the state-space matrices and adding the damping Example 12.3. The resulting eigenvalues show that we are close to achieving critical damping, with just rate feedback, along all three axes. We also show what linear quadratic output feedback achieves where the measurement matrix is just for the rates.

In both cases, the control is the desired angular acceleration. The simple rate controller works reasonably well. We could add gains to cancel the cross-axis terms to decouple the three axes completely. The full-state feedback controller has very low gains on an angle. There are small cross-axis gains. The full-state feedback controller has the same frequencies for all three axes. The controller increases the frequency of

```
%% Response of a gravity gradient system

iX = 0.2; iY = 0.4; iZ = 0.1;
n  = sqrt(3.98600436e5/7000^3);
kX = 4*n^2*(iY-iZ)/iX;
kY = 3*n^2*(iX-iZ)/iY;
kZ =   n^2*(iX-iZ)/iZ;
cX =  n*(iY-iZ-iX)/iX;
cZ =  n*(iZ+iX-iY)/iZ;
a  = [zeros(3,3) eye(3);...
      -kX 0 0  0  0 -cX;...
      0 -kY 0  0  0 0;...
      0  0 -kZ -cZ 0 0];
eig(a)
zeta         = 0.7071;
u            = -2*zeta*diag(sqrt([kX kY kZ]));
a(4:6,4:6)   = a(4:6,4:6) + u;
eig(a)
c = [zeros(3,3) eye(3)];
b  = [zeros(3,3);eye(3)];
k  = QCR( a, b, eye(6), 1e10*eye(3) )
eig(a-b*k)
```

```
Undamped
-0.0000 ± 0.0028i
0.0000 ± 0.0010i
0.0000 ± 0.0009i

Damped
-0.0019 ± 0.0021i
-0.0007 ± 0.0007i
-0.0007 ± 0.0007i

Full State Feedback Gain Matrix
5.1260e-06  3.8755e-22  1.2362e-06  1.2305e-03  4.3687e-19  2.3552e-04  -
4.3567e-22  9.1663e-06  2.9010e-21  -9.7730e-21  3.1603e-03  6.7100e-19  -
1.4969e-06 1.6850e-21 8.8290e-06 2.3552e-04 5.1413e-19 2.9109e-03
Full State
-0.0025 ± 0.0028i
-0.0022 ± 0.0020i
-0.0022 ± 0.0022i
```

Example 12.3: Gravity gradient.

the pitch and yaw eigenvalues, that is it is adding stiffness to the system. Note that the cross-axis angular gains are of the same order of magnitude as the main diagonal terms. The cost of the control was adjusted so that the roll frequency is about the same as the rate damping case. You cannot just zero the angle gains, i.e., not measure angles. Interestingly, if you do you get four poles with zero imaginary parts and one complex pair.

One could also try output feedback assuming that we could measure pitch and roll and not yaw. Roll and pitch can be measured by an Earth sensor.

We have not specified how we would do the damping. Actuation could be done by rotational actuators, like reaction wheels, or by linear actuators aligned with the three axes and offset from the center-of-mass.

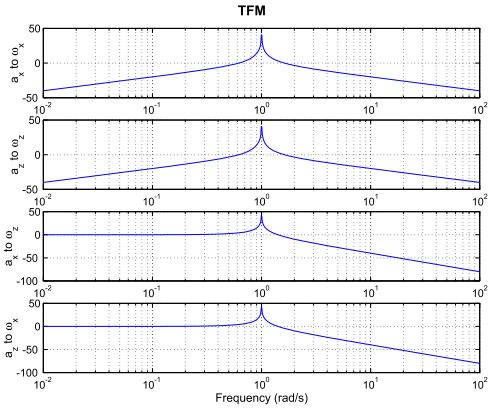
12.8. Nutation control

Nutation control is applicable to spinning spacecraft and those with momentum wheels. A step in angular acceleration in x will produce an offset in rate in z but not in x. A step in angular acceleration in z will produce an offset in rate in x but not in z. There is an undamped pole but the only zeros are at zero frequency.

Damping can be provided by setting $a_x = -2\zeta k\omega_x$. The transfer function becomes

$$\begin{bmatrix} s+2\zeta k & k \\ -k & s \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_z \end{bmatrix} = \begin{bmatrix} a_x \\ a_z \end{bmatrix} \tag{12.29}$$

```
k = 1;
a = [0 k;-k 0];
b = eye(2);
c = eye(2);
d = zeros(2,2);
w = logspace(-2,2,1000);
[m(1,:),p(1,:)] = Fresp(a,b,c,d,1,1,w);
[m(2,:),p(2,:)] = Fresp(a,b,c,d,2,2,w);
[m(3,:),p(3,:)] = Fresp(a,b,c,d,1,2,w);
[m(4,:),p(4,:)] = Fresp(a,b,c,d,2,1,w);
Plot2D(w,20*log10(m),'Frequency (rad/s)',...
       {'a_x to \omega_x' 'a_z to \omega_z' 'a_x to \omega_z' 'a_z to \omega_x'},'TFM','xlog')
PrintFig(1,1,1,'NutationDampingTFM')
```



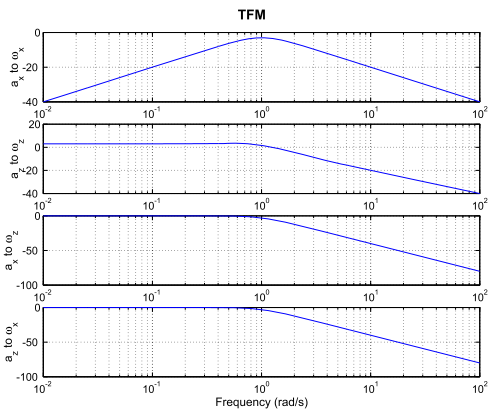
Example 12.4: Nutation dynamics.

The resulting solution is

$$\begin{bmatrix} \omega_x \\ \omega_z \end{bmatrix} = \frac{\begin{bmatrix} s & -k \\ k & s+2\zeta k \end{bmatrix}}{s^2+2\zeta ks+k^2} \begin{bmatrix} a_x \\ a_z \end{bmatrix} \tag{12.30}$$

The system is now critically damped at the expense of allowing a step response in the z-axis to produce a rate offset in z. An alternative is to set $a_x = -\zeta k\omega_x$ and $a_z = -\zeta k\omega_z$. This has the added advantage of tolerating a failure in either channel.

```
k = 1;
zeta = 0.7071;
a = [-2*zeta*k k;-k 0];
b = eye(2);
c = eye(2);
d = zeros(2,2);
w = logspace(-2,2,1000);
[m(1,:),p(1,:)] = Fresp(a,b,c,d,1,1,w);
[m(2,:),p(2,:)] = Fresp(a,b,c,d,2,2,w);
[m(3,:),p(3,:)] = Fresp(a,b,c,d,1,2,w);
[m(4,:),p(4,:)] = Fresp(a,b,c,d,2,1,w);
Plot2D(w,20*log10(m),'Frequency (rad/s)',...
       {'a_x to \omega_x' 'a_z to \omega_z' 'a_x to \omega_z' 'a_z to \omega_x'},'TFM','xlog')
PrintFig(1,1,1,'NutationDampingTFM2')
```



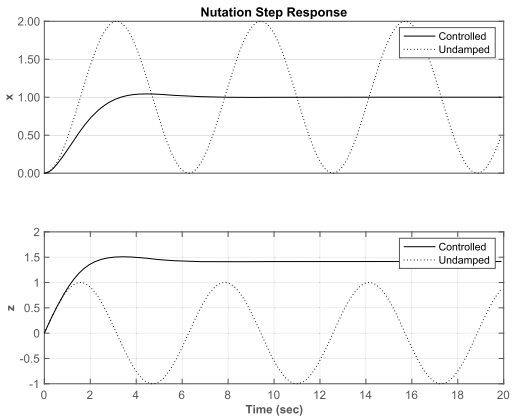
Example 12.5: Nutation dynamics with a rate damper.

A step in angular acceleration in x will produce an offset in rate in z but not in x. A step in angular acceleration in z will produce an offset in rate in x but not in z.

There is an undamped pole but the only zeros are at zero frequency. If the dual-channel approach is used there will be offsets in both axes.

Example 12.6 shows the step response of the damped and undamped system. Note the offset in z in the damped system.

```
k       = 1;
zeta    = 0.7071;
a       = [-2*zeta*k k;-k 0];
b       = eye(2);
c       = eye(2);
d       = zeros(2,2);
g       = statespace(a,b,c,d);
y       = Step(g,2,0.1,200);

k       = 1;
zeta    = 0;
a       = [2*zeta*k k;-k 0];
g       = statespace(a,b,c,d);
[y2,x,t] = Step(g,2,0.1,200);
lL      = {'Controlled' 'Undamped'};

TimeHistory(t,[y;y2], {'x' 'z'},...
      'Nutation Step Response', {[1 3] [2 4]},{
            lL lL},1)
```



Example 12.6: Step response.

12.9. Momentum-bias Earth-pointing control

The roll/yaw and pitch dynamics can be decoupled for analysis of normal operations. The pitch dynamics are a double integrator. The roll dynamics are a coupled fourth-order plant. There are two pure imaginary pole pairs. The first at orbit rate is due to the kinematics, the second is the nutation mode, and is due to the bias momentum. The main diagonal channels each have a pure imaginary zero pair. This zero pair is located between the orbit rate and nutation mode providing 180 degrees of phase shift and making the control problem easier.

Disturbances at orbit rate, or the nutation frequency, will cause uncontrolled growth in the attitude errors. Since many of the disturbance sources are at orbit rate, it is necessary to add an automatic control system. The nutation pole causes large oscillatory responses whenever a sudden torque is applied to the spacecraft.

The high- and low-frequency limits are of interest. At high frequencies as s approaches infinity, the system becomes a double integration. At low frequencies, the orbit-rate pole pair dominates.

The attitude kinematics come from the small-angle approximation

$$\omega = \dot{\theta} + (1 - \theta^{\times})\nu \tag{12.31}$$

where ν is the orbit rate vector. The dynamical equations are

$$I\dot{\omega} + \omega^{\times}(I\omega + h_W) + \dot{h}_W = T \tag{12.32}$$

For the purpose of analyzing the roll/yaw dynamics assume that h_W is constant. The orbit-rate vector is $[0 \quad -\omega_o \quad 0]^T$. Assume also that the inertia matrix is diagonal. Linearizing the dynamics, and coupling with the kinematics gives the transfer-function matrix

$$\begin{bmatrix} \theta_x(s) \\ \theta_y(s) \end{bmatrix} = \frac{\begin{bmatrix} I_z s^2 + \omega_o h_z & (\omega_o I_x - h_x)s \\ -(\omega_o I_z - h_z)s & I_x s^2 + \omega_o h_x \end{bmatrix}}{I_x I_z (s^2 + \omega_o^2)\left(s^2 + \left(\frac{h_x h_z}{I_x I_z}\right)\right)} \begin{bmatrix} T_x/I_x \\ T_x/I_x \end{bmatrix} \tag{12.33}$$

$$h_x = h_w + \omega_o(I_y - I_z)$$
$$h_z = h_w + \omega_o(I_y - I_x)$$

The roll/yaw equations and the equation of the damper wheel are

$$\begin{aligned} I_x\dot{\omega}_x + \omega_y\omega_z(I_z - I_y) - \omega_z h_w + J(\dot{\Omega} + \dot{\omega}_x) &= T_x \\ I_z\dot{\omega}_z + \omega_y\omega_x(I_y - I_x) - \omega_y(\Omega + \omega) + \omega_x h_w &= T_z \\ J(\dot{\Omega} + \dot{\omega}_x) &= -D\Omega \end{aligned} \tag{12.34}$$

where J is the inertia of the damper wheel and D is the damping coefficient. If we lump J with I_x we can simplify the above to

$$\begin{aligned} \dot{\omega}_x + k_x\omega_z + \epsilon\dot{\Omega} &= 0 \\ \dot{\omega}_z + k_z\omega_x - \gamma\dot{\Omega} &= 0 \\ \dot{\Omega} + \dot{\omega}_x &= \sigma\Omega \end{aligned} \tag{12.35}$$

where

$$k_x = \frac{\omega_y(I_z - I_y) - h_w}{I_x} \tag{12.36}$$

$$k_z = \frac{\omega_y(I_y - I_x) + h_w}{I_z} \tag{12.37}$$

$$\gamma = \frac{\omega_y J}{I_z} \tag{12.38}$$

$$\sigma = \frac{D}{J} \tag{12.39}$$

$$\epsilon = \frac{J}{I_x} \tag{12.40}$$

The characteristic equation can be written in Evans form

$$\frac{s^2 - \left(\frac{k_x(k_z+\gamma)}{1-\epsilon}\right)}{s^2 - k_x k_z} = -\frac{\sigma}{1-\epsilon} \tag{12.41}$$

When $\sigma = 0$ the numerator polynomial gives the poles of the system. When $\sigma = \infty$ the denominator gives the poles of the system. Hence, the open-loop zeros are given by the denominator and the open-loop poles by the numerator. The $s = 0$ pole is the damper-wheel pole and the quadratic terms give the roll/yaw poles. For stability, the open-loop zeros must be between the closed-loop poles. These equations are only valid for $J \neq 0$.

$$\begin{bmatrix} \theta_x(s) \\ \theta_y(s) \end{bmatrix} = \frac{\begin{bmatrix} 1/I_x & 0 \\ 0 & 1/I_z \end{bmatrix}}{s^2} \begin{bmatrix} T_x \\ T_z \end{bmatrix} \tag{12.42}$$

As is evident from the transfer-function matrix, high-bandwidth controllers can ignore the orbit rate and nutational dynamics. At low frequencies the plant becomes

$$\begin{bmatrix} \theta_x(s) \\ \theta_y(s) \end{bmatrix} = \frac{\begin{bmatrix} \omega_o & (\omega_o I_x/h_x - 1)s \\ (1 - \omega_o I_z/h_z)s & \omega_o \end{bmatrix}}{s^2 + \omega_o^2} \begin{bmatrix} T_x/h_x \\ T_z/h_z \end{bmatrix} \tag{12.43}$$

The diagonal terms are small and the dynamics are dominated by crosscoupling torques. Controllers concerned with attenuating low frequency (near orbit rate) can ignore the nutation mode.

Since there will always be an external bias torque on the spacecraft due to external disturbances, which causes the wheels to accumulate momentum over time, the wheels must be unloaded. To unload a wheel is to reduce the average speed back to its nominal value. This can be done with any method that also produces an external torque, i.e., magnetic torquers, reflective surfaces, aerodynamic surfaces, or thrusters. An alternative is to partition the control effort so that the inertially fixed disturbance compensation is performed by thrusters or torquers, and only the cyclic disturbances are controlled by the wheels. This can be done with a complementary filter pair – a matched low- and high-pass filter pair such as

$$\frac{u_{\text{wheel}}}{u} = \frac{s}{s+a} \qquad \frac{u_{\text{thruster}}}{u} = \frac{a}{s+a} \tag{12.44}$$

where a is the break frequency and $s = j\omega$. Choosing a lower break frequency will mean the wheels store more momentum, while choosing a higher frequency will cause the thrusters to fire more often. The value must be selected so that the wheels never saturate over the necessary period.

Reaction wheels must pass through zero wheel speed; this presents a problem because the internal friction torques have a discontinuity, shown in Fig. 12.6. This is highly

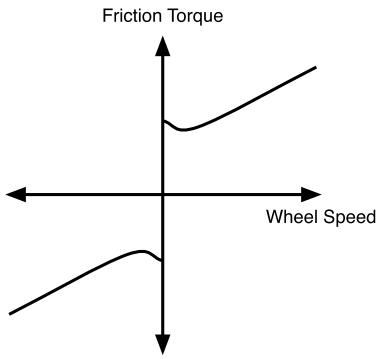


Figure 12.6 Friction-torque diagram.

nonlinear and not amenable to linearization about the zero wheel speed! The wheel can also get stuck at zero speed. One standard approach is to use triangle-wave dither in the voltage input to the wheel motor. In this technique, a triangle voltage wave is added to the desired voltage command. The command to the wheels is the sum of the triangle wave and the commands. The triangle-wave amplitude is a few least-significant bits. The effect of dither and other methods of improving control resolution is shown in Example 12.7. A PD controller is designed using the toolbox function PDDesign and discretized using a zero-order hold (C2DZOH). The ideal step response is shown in blue and labeled "no minimum pulsewidth" in the legend, along with three methods of handling finite pulsewidth.

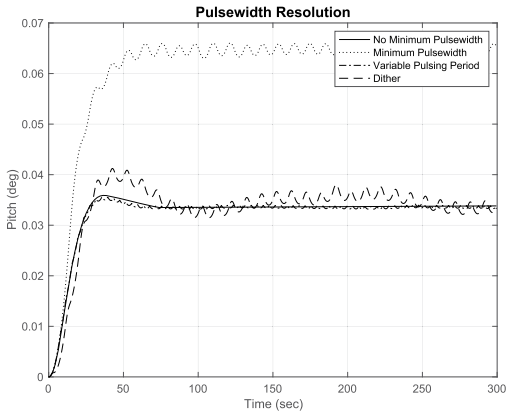
The best result is obtained by using a different output period from the period used for control. For example, the algorithm might run at 2 Hz but the output is only sampled at 0.25 Hz. Triangle dither also works well. Another approach is to embed the wheel in a tachometer loop where the outer loop commands wheel-speed changes and the inner loop tries to force the wheel speed to track the desired speed changes. This tachometer loop is in addition to the current feedback loop, used in most reaction wheels to control the back EMF.

12.10. Mixed control

The spacecraft can be controlled using fewer than three wheels, if the magnetic torquers are used to supplement the wheels. The relationship between torque and dipole is

$$T = m \times b = -b \times m \tag{12.45}$$

```
inr = 2000; dT = 0.5;
minPw = 20; mDither = 25;% milliseconds
nPulse = 8; wDither = 30;
zeta = 0.7071; wN = 0.1; wD = 1.0;
[a,b,c,d]      = PDDesign(zeta,wN,wD,inr);
[aN,bN,cN,dN] = CButter(2,1);
[aC,bC,cC,dC] = Series(a,b,c,d,aN,bN,cN,dN);
[aC,bC]        = C2DZOH( aC, bC, dT );
aP             = [0,1;0 0];
bP             = [0;1]/inr;
[aP, bP]       = C2DZOH( aP, bP, dT/500);

nSim = 600; uStep = 0.01;
yPlot = zeros(4,nSim); uPlot = yPlot;
dither = mDither*TriangleWave(linspace(0,2*pi*
       wDither,nSim));
uTorque = 0.5; nPW = 1000*dT;

for j = 1:4
  x       = zeros(2,1);
  xC      = zeros(3,1);
  kPulse = 0;
  for k = 1:nSim

    yPlot(j,k) = x(1);
    uC = cC*xC + dC*x(1);
    xC = aC*xC + bC*x(1);
    pulsewidth = dT*abs(uC/uTorque)*1000;

    if( j == 3 )
      kPulse = kPulse + 1;
      if( kPulse < nPulse )
        pulsewidth = 0;
      else
        pulsewidth = nPulse*pulsewidth;
        kPulse     = 0;
      end
    elseif( j == 4 )
      pulsewidth = pulsewidth + dither(k);
    end

    if( (j > 1) & (pulsewidth < minPw) )
      pulsewidth = 0;
    end

    uPlot(j,k) = abs(pulsewidth);
    nPulsewidth = floor(pulsewidth);
    uCA = [uTorque*sign(uC)*ones(1,nPulsewidth),
           zeros(1,nPW-nPulsewidth)];
    for i = 1:nPW
      x = aP*x + bP*(uStep - uCA(i));
    end
  end
end

Plot2D((0:(nSim-1))*dT,yPlot*180/pi,...
       'Time (sec)','Pitch (deg)',...
       'Pulsewidth Resolution');
legend('No Minimum Pulsewidth',...
       'Minimum Pulsewidth',...
       'Variable Pulsing Period',...
       'Dither');
PrintFig(1,1,1,'Pulsewidth')
```



Example 12.7: Control resolution example showing multiple ways of handling finite pulsewidth.

or

$$T = \begin{bmatrix} 0 & b_z & -b_y \\ -b_z & 0 & b_x \\ b_y & -b_x & 0 \end{bmatrix} m \tag{12.46}$$

The torquers can produce an exact two-axis torque

$$m = \text{pinv}\left(a(k,:)\right) T(k) \tag{12.47}$$

where $b(k,:)$ are the rows of T that you want to produce and pinv is the pseudoinverse and $a = -b$. This allows one reaction wheel to be operated with the three magnetic torquers to produce a given 3-axis control torque.

A single-axis torque can be realized with two torquers. For example, write the equation for the y-axis.

$$T_y = m_z b_x - m_x b_z \tag{12.48}$$

Then,

$$m_z = \frac{T_y + m_x b_z}{b_x} \tag{12.49}$$

Find the dipoles that minimizes

$$m_x^2 + m_z^2 \tag{12.50}$$

This is found by replacing m_z in the quadratic, taking the derivative and solving for the dipole

$$m_x = -\frac{b_z T_y}{b_x^2 + b_z^2} \tag{12.51}$$

12.11. Magnetic-torquer-only control

Magnetic torquers are simple, and relatively inexpensive attitude control devices. The following sections discuss ways to use magnetic torquers.

12.11.1 BDot

BDot, which stands for magnetic-field derivative, is a way of damping angular rates without a gyro. The magnetic field can be measured by a magnetometer in the body frame. Let the transformation matrix from the inertial to the body frame be A. The transformation is

$$b_b = A b_i \tag{12.52}$$

The derivative (assuming that the field is not changing in the inertial frame) is

$$\dot{A} = A\omega^{\times} \tag{12.53}$$

Therefore

$$\dot{b}_b = \omega^{\times} b_b \tag{12.54}$$

The torque due to a magnetic torquer is

$$T = M^{\times} b_b \tag{12.55}$$

For a damper we want the torque to be

$$T = -k\omega \tag{12.56}$$

We can make a controller of the form

$$M = k\omega \dot{b} \tag{12.57}$$

which results in a controller

$$T = -k b_b^{\times} b_b^{\times} \omega \tag{12.58}$$

The controller in the s-plane is

$$M = skb \tag{12.59}$$

where s is the derivative, b is the magnetic field, and k is the gain. It is a good idea to provide some filtering. A first-order filter can be added

$$M = \frac{skb}{\tau_f s + 1} \tag{12.60}$$

This can be made into a discrete-time controller by using the first different operator

$$s = \frac{1 - z^{-1}}{\tau} \tag{12.61}$$

where τ is the time step. The resulting derivative is

$$\dot{b}_k = \frac{\tau \dot{b}_{k-1} + b_k - b_{k-1}}{\tau + \tau_f} \tag{12.62}$$

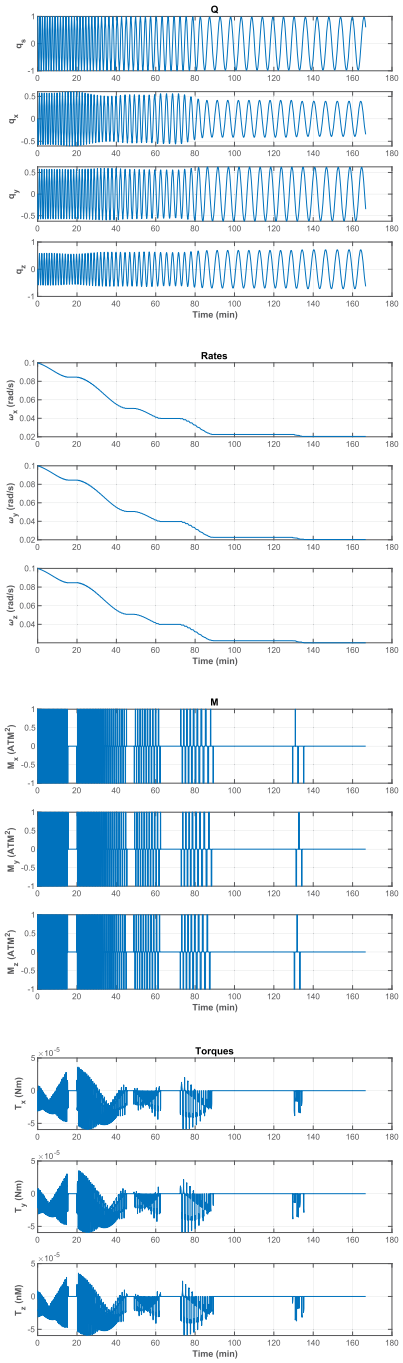
Example 12.8 shows the damping response to an initial body rate.

12.12. Low-bandwidth small-angle control

We will design a controller for a momentum-bias satellite in low-Earth orbit. It is Earth pointing. The control system is shown in Fig. 12.7. The momentum wheel controls pitch using a PD controller. Roll and yaw are controlled using state feedback. The angular momentum is

$$H = A\left(I\omega + uJ(\Omega + \omega)\right) \tag{12.63}$$

```
secInDay          = 86400;

% Parameters
inertia = eye(3);
dT = 1;
gain = 10;
mMax = 1;
mMin = 1e-6;
el = [7000;pi/3;0;0;0;0];
jD0 = Date2JD([2024 4 4]);
tEnd = 10000; % Duration of the simulation
tauFilter         = 10; % s
omega = [0.1;0.1;0.1];
q = [1;0;0;0];

%% End of User Inputs
n = ceil(tEnd/dT);              % Time steps
t = linspace(0,tEnd,n);    % Time array
jD = jD0 + t/secInDay;      % Julian date array
dRHS.inertia = inertia;
dRHS.mu = 3.98600436e5;
% Orbit
[r,v] = El2RV(el);
% Plotting arrays
xP = zeros(19,n);
% Simulation loop
x = [r;v;q;omega];
% For the control
bBodyOld = QForm( x(7:10),BDipole(x(1:3),jD(1)));
bDotOld = [0;0;0];
for k = 1:n
  % ECI to body quaternion
  qECIToBody = x(7:10);
  % Magnetic field
  bECI        = BDipole(x(1:3),jD(k));
  bBody       = QForm(qECIToBody,bECI);
  % Low pass filter
  deltaB      = bBody-bBodyOld;
  bDot        = (dT*bDotOld + deltaB)/(tauFilter +
        dT);
  bDotOld     = bDot;
  bBodyOld    = bBody;
  dipole      = gain*bDot;
  j           = find(abs(dipole)<mMin);
  dipole(j) = 0;

  dipole          = -mMax*sign(dipole);
  dRHS.torque     =  Cross(dipole,bBody);

  % Plotting storage
  xP(:,k) = [x;dipole;dRHS.torque];

  % Propagate the state
  x   = RK4(@RHS,x,dT,0,dRHS);
end

% Plot labels
yL       = {'r_x (km)' 'r_y (km)' 'r_z (km)'   ...
 'v_x (km/s)' 'v_y (km/s)' 'v_z (km/s)' ...
 'q_s' 'q_x' 'q_y' 'q_z' ...
 '\omega_x (rad/s)' '\omega_y (rad/s)' '\omega_z
       (rad/s)'...
 'M_x (ATM^2)' 'M_y (ATM^2)' 'M_z (ATM^2)'...
 'T_x (Nm)' 'T_y (Nm)' 'T_z (nM)'};

% Plot
k = 7:10;
TimeHistory(t,xP(k,:),yL(k),'Q'); k = 11:13;
TimeHistory(t,xP(k,:),yL(k),'Rates'); k = 14:16;
TimeHistory(t,xP(k,:),yL(k),'M'); k = 17:19;
TimeHistory(t,xP(k,:),yL(k),'Torques');
Figui

%% Right hand side function
function xDot = RHS(x,~,d)

r      = x( 1: 3);
v      = x( 4: 6);
q      = x( 7:10);
omega = x(11:13);

omegaDot = d.inertia\(d.torque - Cross(omega,d.
       inertia*omega));

xDot = [v;-d.mu*r/Mag(r)^3;QIToBDot(q,omega);
       omegaDot];

end
```



Example 12.8: BDot simulation.

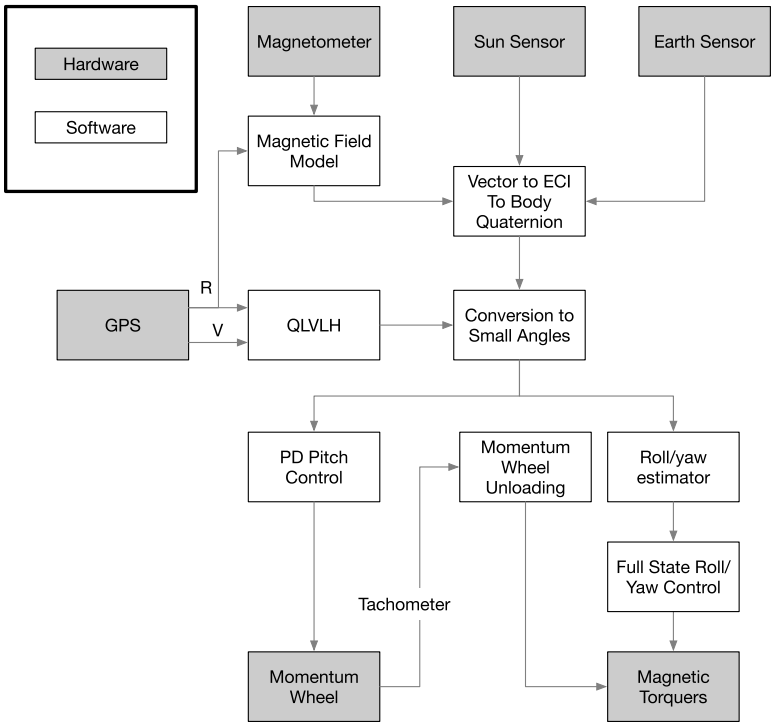


Figure 12.7 Low-bandwidth control system.

where u is the wheel spin-rate unit vector and J is the momentum of inertia about that axis. Ω is the angular rate of the wheel relative to the core spacecraft. The equations of motion are found by taking the time derivative in the inertial frame.

$$T = I\dot{\omega} + uJ(\dot{\Omega} + \dot{\omega}) + \omega^{\times}\left(I\omega + uJ(\Omega + \omega)\right) \tag{12.64}$$

$$T_w = J(\dot{\Omega} + \dot{\omega}) \tag{12.65}$$

These can be further simplified,

$$T = I\dot{\omega} + uT_w + \omega^{\times}h_w \tag{12.66}$$

$$T_w = J(\dot{\Omega} + \dot{\omega}) \tag{12.67}$$

If the angles are small, the transformation matrix from LVLH to body is

$$A = \begin{bmatrix} 1 & \theta_z & -\theta_y \\ -\theta_z & 1 & \theta_x \\ \theta_y & -\theta_x & 1 \end{bmatrix} \tag{12.68}$$

θ_x about x, θ_y about y, and θ_z about z. The angles and signs can be found by inspection by looking at Fig. 12.8.

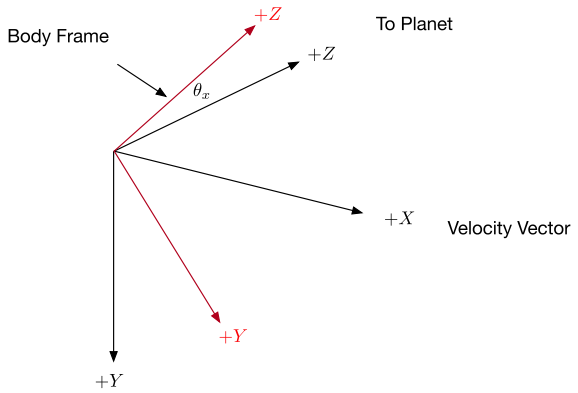


Figure 12.8 Coordinate transformations from LVLH to body with small angles. This diagram shows the roll angle, α. The body axes are in red.

Assume that the spacecraft inertial rotation rate is ω_o about $-Y$, then the angular rates in the body frame are

$$\omega = \begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \end{bmatrix} + \begin{bmatrix} 1 & \theta_z & -\theta_y \\ -\theta_z & 1 & \theta_x \\ \theta_y & -\theta_x & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -\omega_o \\ 0 \end{bmatrix} \tag{12.69}$$

The dynamical equations are

$$I_x\dot{\omega}_x + h_w\omega_z = T_x \tag{12.70}$$

$$I_y\dot{\omega}_y = T_y - T_w \tag{12.71}$$

$$I_z\dot{\omega}_z - h_w\omega_x = T_z \tag{12.72}$$

The system is linear. The state equations are

$$\begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \\ \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\omega_o & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \omega_o & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \frac{h_w}{I_x} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{h_w}{I_z} & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{12.73}$$

$$+\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{I_x} & 0 & 0 \\ 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & \frac{1}{I_z} \end{bmatrix} \begin{bmatrix} T_x \\ T_y - T_w \\ T_z \end{bmatrix}$$

The roll/yaw state equations are

$$\begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_z \\ \dot{\omega}_x \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} 0 & -\omega_o & 1 & 0 \\ \omega_o & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{h_w}{I_x} \\ 0 & 0 & -\frac{h_w}{I_z} & 0 \end{bmatrix} \begin{bmatrix} \theta_x \\ \theta_z \\ \omega_x \\ \omega_z \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{I_x} & 0 \\ 0 & \frac{1}{I_z} \end{bmatrix} \begin{bmatrix} T_x \\ T_z \end{bmatrix} \tag{12.74}$$

The controller uses full-state feedback and is

$$T = -kx \tag{12.75}$$

Generally, angular rates are not measured directly. An easy way to find the rates, given the model, is to make a full-state estimator. The estimator will be of the form

$$\dot{x}_E = Ax_E + k_E(y - Cx_E) \tag{12.76}$$

where k_E is the optimal estimator gain, y is the measurement vector, and C is the measurement matrix

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{12.77}$$

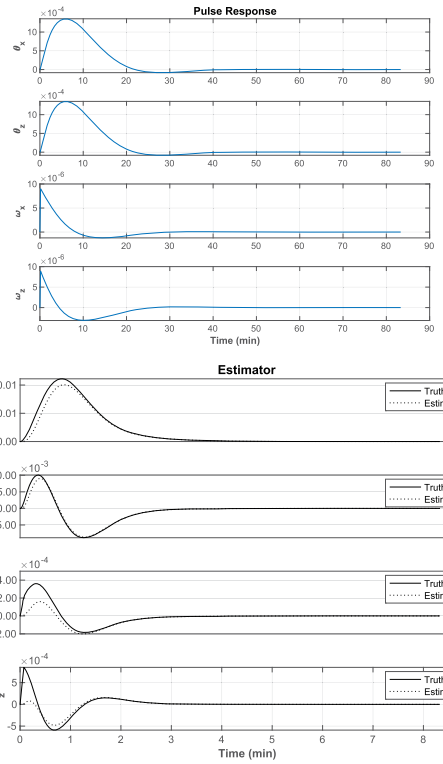
The measurement vector is

$$y = \begin{bmatrix} \theta_x \\ \theta_z \end{bmatrix} \tag{12.78}$$

The parameters for the satellite, a 3U CubeSat, are given in Table 12.1.

Example 12.9 shows the control response. The script allows you to explore different control options. Note that the estimator response tracks the truth values very well, even though the estimator does not include the control torque. You can try stabilizing the system with only one control torque. While you can stabilize it, you cannot control both axes.

```
oneTorque = false; % Pick option to use one
       torque

omegaO     = 2*pi/(60*90);
hW         = 0.0005;

iX = 0.025; iY = .025; iZ = 0.005;

% Full system
a = [0       -omegaO 1       0;...
     omegaO  0       0       1;...
     0       0       0       hW/iX;...
     0       0       -hW/iZ  0];

b = [0 0;0 0;1/iX 0;0 1/iZ];
c = [1 0 0 0;0 1 0 0];

if( oneTorque )
  bC = [0;0;0;1/iZ]; %#ok<UNRCH>
  r  = 1e8;
else
  bC = b;
  r  = 1e10*eye(2);
end

disp('Open Loop Eigenvalues')
eig(a)

kC = QCR(a,bC,diag([1 1 0.1 0.1]),r);

kE = QCE( a, c, 1e-10*eye(2), 0.1*eye(2), b );

% Controller gain
disp(kC);

% Estimator gain
disp(kE);

% Closed loop eigenvalues
aCL = a-bC*kC;
disp('Closed Loop Eigenvalues')
eig(aCL)

% Pulse response
dT       = 0.5;
jP       = 10;
[aD,bD] = C2DZOH(a,b,dT);
[~,bCD] = C2DZOH(a,bC,dT);
tP       = [1e-6;1e-6];

n        = 1000;
x        = zeros(4,n);
xE       = zeros(4,n);
for j = 2:n
  xEJM1 = xE(:,j-1);
  %xEJM1 = x(:,j-1);
  if( j <= jP )
    x(:,j) = aD*x(:,j-1) - bC*kC*xEJM1 + bD*tP;
  else
    x(:,j) = aD*x(:,j-1) - bC*kC*xEJM1;
  end
  xE(:,j) = aD*xEJM1 + kE*(x(1:2,j-1)-c*xEJM1) -
        bC*kC*xEJM1;
end

t = (0:n-1)*dT;

yL = {'\theta_x' '\theta_z'...
      '\omega_x' '\omega_z'};
TimeHistory(t,x,yL,'Pulse Response')
lL = {'Truth' 'Estimate'} ;
leg = {lL lL lL lL };
j   = {[1 5] [2 6] [3 7] [4 8]};
TimeHistory(t,[x;xE],yL,'Estimator',j,leg);
```



Example 12.9: Low-bandwidth control simulation.

Table 12.1 Spacecraft parameters.

Parameter	Value
Mass	3 kg
I	$\begin{bmatrix} 0.025 & 0 & 0 \\ 0 & 0.025 & 0 \\ 0 & 0 & 0.005 \end{bmatrix}$ kg m^2
h_w	0.04 N m s
ω_0	0.0011 rad/s

The next step is to use magnetic torquers. For this example, we assume that we only want to use the torquers for roll and yaw. We have three degrees-of-freedom, but only two torques to match. The approach is to pick the three torquer commands so that the roll and yaw torques are met exactly and the third torque is minimized. The cost function is

$$J = (T - M^{\times} b)^T (TM^{\times} b) + M^T M \tag{12.79}$$

with two elements constrained to exactly match the desired torques. The algorithm is

$$s = b^{\times} \tag{12.80}$$

$$c = ss \tag{12.81}$$

$$\lambda = -\frac{T}{c(a, a)} \tag{12.82}$$

$$M = s(1:3, a)\lambda \tag{12.83}$$

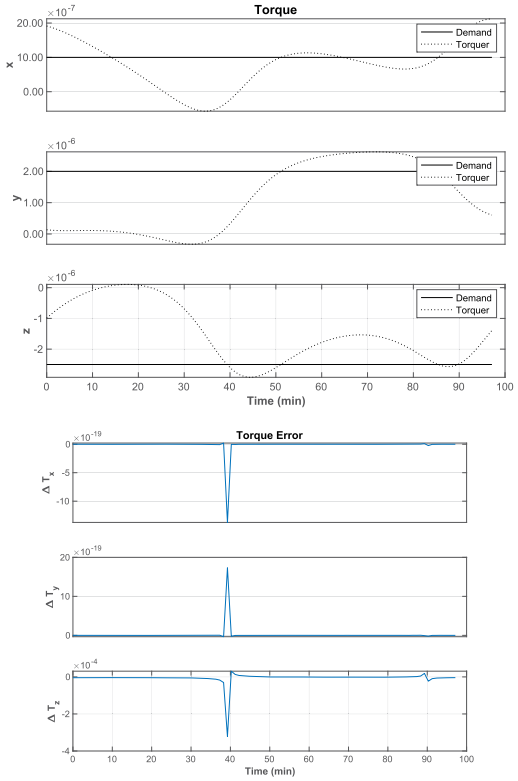
where a is an index array with two elements.

If you wanted to also control pitch, this can only be done approximately using an algorithm such as

$$M = \frac{b^{\times} T}{|b|^2} \tag{12.84}$$

where b is the magnetic field and T is the desired torque, both in the body frame. The performance will be very dependent on the orbit. Example 12.10 shows the results for the two methods in a 28.6-deg orbit. The spike in torque error is due to the 2×2 matrix being ill-conditioned. This does not, at least in this orbit, pose a particular problem in computing the dipole.

```
a        = 7000;
e        = 0;
i        = 28.4*pi/180;
jD0      = Date2JD([4 4 2030]);

[r,v,t] = RVOrbGen([a i 0 0 e 0]);
jD       = jD0 + t/86400;

b        = BDipole(r,jD);
n        = length(t);

b        = QForm(QLVLH(r,v),b);

tD       = [1e-6;2e-6;-2.5e-6].*ones(1,n);
tM       = zeros(3,n);
m        = zeros(3,n);
for k = 1:n
  m(:,k)  = Skew(b(:,k))*tD(:,k)/Mag(b(:,k))^2;
  tM(:,k) = Skew(m(:,k))*b(:,k);
end

lL = {'Demand' 'Torquer'} ;
leg = {lL lL lL };
j   = {[1 4] [2 5] [3 6]};
yL  = {'x' 'y' 'z'};
TimeHistory(t,[tD;tM],yL,'Torque',j,leg);

% Pick x and y
a = [1;2];

for k = 1:n
  s       = Skew(b(:,k));
  c       = s*s;
  cA      = c(a,a);
  lambda  = -cA\tD(a,k);
  m(:,k)  = s(:,a)*lambda;
  tM(:,k) = Skew(m(:,k))*b(:,k);
end

yL  = {'\Delta T_x' '\Delta T_y' '\Delta T_z'};
TimeHistory(t,tD-tM,yL,'Torque Error');
```



Example 12.10: Torque generation for magnetic torquers.

12.13. Lyapunov control

This section will use Lyapunov's direct method to control the nonlinear attitude dynamics and kinematics. The dynamical system is

$$T = I_c\dot{\omega} + \omega^{\times}\left(I_c\omega + I_w(\omega_w + \omega)\right) + J(\dot{\omega}_w + \dot{\omega}) \tag{12.85}$$

$$T_w = I_w(\dot{\omega}_w + \dot{\omega}) \tag{12.86}$$

$$q = f(q, \omega) \tag{12.87}$$

where q is the quaternion, ω is the angular rate of the core spacecraft, ω_w is the wheel angular rates with respect to the spacecraft core, T_w is the wheel torque, I_w is the wheel inertia, T is the external torque, and I_c is the inertia matrix for the core. The quaternion transforms from the inertial to the body frame. Three orthogonal wheels are used. This can be simplified if we let $I = I_c + I_w$.

$$T = I\dot{\omega} + \omega^{\times}I\omega + T_w + \omega^{\times}h_w \tag{12.88}$$

$$T_w = I_w(\dot{\omega}_w + \dot{\omega}) \tag{12.89}$$

$$q = f(q, \omega) \tag{12.90}$$

where $h_w = I_w\omega_w$. Lyapunov's direct method has two steps. The first is to find a scalar function of the states of the system, known as the Lyapunov function, and then to evaluate its derivative along the trajectory of the system. If the Lyapunov function is always decreasing the system will eventually reach equilibrium.

The controller will try to drive the body rates and the quaternion error to zero. Define the error quaternion as

$$q_E = q^T \otimes q_T \tag{12.91}$$

where q_T is the target quaternion. The error quaternion consists of a scalar and vector part,

$$q_E = \begin{bmatrix} s \\ u \end{bmatrix} \tag{12.92}$$

where u is a 3-by-1 vector and s is a scalar.

If there exists a Lyapunov function $V(x)$, where x is the state vector that is to be controlled, such that

1. $V(x)$ is positive-definite;
2. $\dot{V}(x)$ is negative-definite;
3. $V(x) \to \infty$ as $||x|| \to \infty$;

then the equilibrium point is uniformly asymptotically stable. For this dynamical system [1]

$$V = \frac{1}{2}\omega^T I\omega + \eta\left((1-s)^2 + u^T u\right) \tag{12.93}$$

The first time derivative is

$$\dot{V} = \frac{1}{2}\left(\omega^T I\dot{\omega} + \dot{\omega}^T I\omega\right) + \eta\left(2s\dot{s} + \dot{u}^T u + u^T\dot{u}\right) \tag{12.94}$$

or

$$\dot{V} = \omega^T I\dot{\omega} + \eta\left(2s\dot{s} + \dot{u}^T u + u^T\dot{u}\right) \tag{12.95}$$

After much algebra this becomes

$$\dot{V} = \omega^T\left(I\dot{\omega} + \eta u\right) \tag{12.96}$$

The control law is

$$T_w = \xi\omega + \eta u - \omega^{\times} h_W \tag{12.97}$$

Combining this with the dynamical equations results in

$$0 = I\dot{\omega} + \omega^{\times} I\omega + \xi\omega + \eta u \tag{12.98}$$

Substituting into $\dot{V}$

$$\dot{V} = \omega^{T}\left(-\omega^{\times} I\omega + \xi\omega\right) \tag{12.99}$$

Finally, since $\omega^{T}(\omega^{\times} I\omega) = 0$,

$$\dot{V} = -\omega^{T}\xi\omega \leq 0 \tag{12.100}$$

which is negative-semidefinite. This is a PD controller with compensation for the wheel-momentum torque.

The performance of the Lyapunov-maneuver control system is shown in Example 12.11.

12.14. Orbit-transfer maneuver

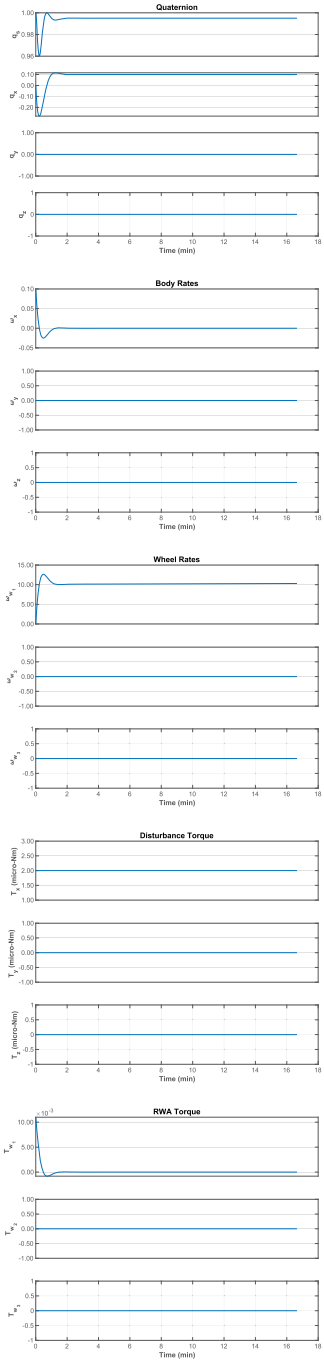
Hohmann transfers are a well-known orbital maneuver used to change the semimajor axis of an orbit.

In this demonstration, we create a 6U CubeSat that has 3 orthogonal reaction wheels and a single hydrazine thruster. The thruster is aligned with the x-axis and must be aligned with the velocity vector to do the maneuver. An ideal Hohmann maneuver is done with impulsive burns at two points in the orbit. In reality, with a thruster, we have to do a finite burn.

The simulation computes burn durations based on the thrust and the mass of the spacecraft. The maneuver is quite small, so the mass change is not important. The Hohmann transfer is computed with the following Spacecraft Control Toolbox code: The first time OrbMnvrHohmann is called it generates the first plot. This shows the initial orbit, the final orbit, and the transfer orbit. The attitude control system uses a proportional-integral-derivative control law. The entire simulation is in a short script. The spacecraft x-axis is aligned with the initial thrust vector so an attitude maneuver is not required. At the final orbit radius, an attitude maneuver is needed to reorient for the final burn.

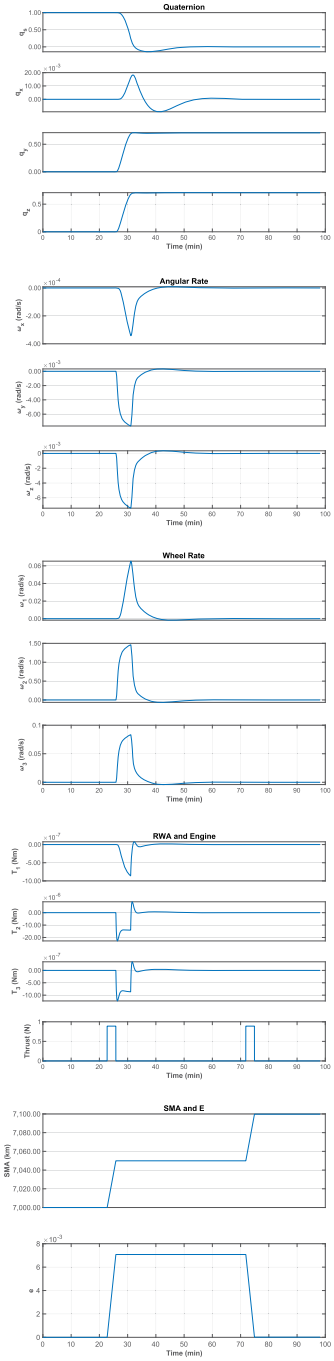
The simulation is shown in Example 12.12. The spacecraft body rates, in the body frame, change only during the reorientation. The reaction-wheel rate changes are proportional to the body-rate changes. The wheel torques are small. The rocket engine is turned on during the "impulsive" burn times. The thruster is a 0.2-lbf hydrazine thruster that is based on the Aerojet-Rocketdyne MR-103. The semimajor axis and eccentricity are shown in the last plot. The middle portion is during the transfer orbit. The eccentricity is zero at the start and finish. Note the slope in both eccentricity and

```
%% Constants
controlOn = true;

%% User initialization
n          = 1000; % Number of simulation steps
dT         = 1.0; % Time step (s)
inertia    = diag([1 2 2]);
qT         = QUnit([1;0.1;0;0]);

% Control
xi = 0.1;
eta = 0.01;

% Initial Julian date
jD0        = Date2JD([2023 4 4]);

% Initial states
q          = [1;0;0;0];
omega      = [0.1;0;0]; % rad/s
omegaRWA   = [0;0;0]; % rad/s
torqueD    = [2;0;0]*1e-6; % Disturbance torque

%% State vector
x     = [q;omega;omegaRWA];
dRHS = RHSGyrostat;
dRHS.inertia = inertia;

%% Simulate
xP         = zeros(16,n);

% Simulation loop
for k = 1:n

   if( controlOn )
     q      = x(1:4);
     omega = x(5:7);
     qE     = QMult(QPose(q),qT);
     tRWA = xi*omega + eta*qE(2:4) - cross(omega,
            diag(dRHS.inrWheel)*x(8:10));
   else
     tRWA = [0;0;0];
   end
   dRHS.torque       = torqueD;
   dRHS.torqueWheel = tRWA;
   xP(:,k)           = [x;torqueD*1e6;tRWA];
   x                 = RK4(@RHSGyrostat,x,dT,0,dRHS
                     );
end

%% Plotting
t  = (0:n-1)*dT;

yL = {'q_s' 'q_x' 'q_y' 'q_z' ...
'\omega_x' '\omega_y' '\omega_z' ...
'\omega_{w_1}' '\omega_{w_2}' '\omega_{w_3}' ...
'T_x (micro-Nm)' 'T_y (micro-Nm)' 'T_z (micro-Nm)'
          ' ...
'T_{w_1}' 'T_{w_2}' 'T_{w_3}'};

tL = {'Quaternion' 'Body Rates' 'Wheel Rates'...
      'Disturbance Torque' 'RWA Torque' };

kL = {1:4 5:7 8:10 11:13 14:16};
for j = 1:length(tL)
   k = kL{j};
   TimeHistory(t,xP(k,:),yL(k),tL{j});
end

Figui
```



Example 12.11: Lyapunov-maneuver control.

```
nToKN     = 0.001;
mu        = Constant('mu earth');
%% Compute the maneuver
rI        = [-7000;0;0];
vI        = [0;-sqrt(mu/Mag(rI));0];
rF        = 7100;
el        = RV2El(rI,vI);
p         = Period(el(1));
OrbMnvrHohmann(Mag(rI),rF);
[dV,tOF]  = OrbMnvrHohmann(Mag(rI),rF);
%% Set up the spacecraft
mP        = 6; % kg
thrustE   = 4.448*0.2; % N
dVTot     = dV.a + dV.b; % km/s
iSp       = 224; % s
fS        = 0.1; % Structural fraction
[mF,mT]   = RocketMass(iSp,mP,fS,dVTot);
inr       = InertiaCubeSat('6U',mT);
acc       = thrustE/mT; % m/s^2
tBA       = dV.a/acc/nToKN; % s
tBB       = dV.b/acc/nToKN; % s
dRHS      = RHSRWAOrbit;
dRHS.inr  = inr;
dRHS.m    = mT;
tStart    = [p/4-tBA/2 p/4+tOF-tBB/2];
tEnd      = tStart + [tBA tBB];

%% Set up the controller
dC                = PID3Axis;
dC.body_vector    = [1;0;0];
dC.mode           = 1; % Align two axes
dC.inertia        = inr;

fprintf('Burn A duration %8.2f s\n',tBA);
fprintf('Burn B duration %8.2f s\n',tBB);
fprintf('Thrust          %8.2f N\n',thrustE);
fprintf('Mass Total      %8.2f kg\n',mT);
fprintf('Mass Fuel       %8.2f kg\n',mF);
fprintf('Initial SMA     %8.2f km\n',el(1));
fprintf('Initial e       %8.2f\n',el(5));

%% Simulation
% ECI burn vector
uBurn  = [1 -1;0 0;0 0];
dT     = 1; % s
n      = ceil(2*tOF/dT);
kMnvr  = 1;
x      = [rI;vI;1;zeros(9,1)];
xP     = zeros(22,n);
inMnvr = false;
t      = (0:n-1)*dT;
% Simulation loop
for k = 1:n
  dC.eci_vector = uBurn(:,kMnvr);
  [tRWA, dC]    = PID3Axis( x(7:10), dC );
  inMnvr = false;
  if( t(k) > tStart(1) && t(k) < tEnd(1) )
    inMnvr = true;
  end
  if( t(k) > tEnd(1) )
    kMnvr = 2;
  end
  if( t(k) > tStart(2) && t(k) < tEnd(2) )
    inMnvr = true;
  end
  if( inMnvr )
    dRHS.force = thrustE*QTForm(x(7:10),dC.
        body_vector)*nToKN; % kN
  else
    dRHS.force = [0;0;0];
  end
  el = RV2El(x(1:3),x(4:6));
  xP(:,k) = [x;tRWA;Mag(dRHS.force)/nToKN;el(1);
      el(5)];
  dRHS.torqueRWA = -tRWA;
  x = RK4(@RHSRWAOrbit,x,dT,0,dRHS);
end
fprintf('Final SMA       %8.2f km\n',el(1));
fprintf('Final e         %8.2f\n',el(5));
yL = {'r_x (km)' 'r_y (km)' 'r_z (km)' ...
'v_x (km/s)' 'v_y (km/s)' 'v_z (km/s)' ...
'q_s' 'q_x' 'q_y' 'q_z' ...
'\omega_x' '\omega_y' '\omega_z' ...
'T_1 (Nm)' 'T_2 (Nm)' 'T_3 (Nm)' ...
'Thrust (N)' 'SMA (km)' 'e'};
k=1:3;TimeHistory(t,xP(k,:),yL(k),'Position');
k=4:6;TimeHistory(t,xP(k,:),yL(k),'Velocity');
k=7:10;TimeHistory(t,xP(k,:),yL(k),'Quaternion');
k=11:13;TimeHistory(t,xP(k,:),yL(k),'Body Rate');
k=14:16;TimeHistory(t,xP(k,:),yL(k),'RWA Rate');
k=17:20;TimeHistory(t,xP(k,:),yL(k),'Actuators');
k=21:22; TimeHistory(t,xP(k,:),yL(k),'SMA and E')
    ;
Figui;
```



Example 12.12: Attitude maneuvers for an orbit transfer.

semimajor axis due to the finite acceleration. The maneuver is not perfect because the burn durations are finite.

The simulation also prints out some results

```
Hohmann Transfer
-----------------------------------------------------
Initial Orbit Radius        =     7000.0000
Final Orbit Radius          =     7100.0000
-----------------------------------------------------
Delta V Total               =        0.0533
Delta V at A                =        0.0267
Delta V at B                =        0.0266
E transfer                  =        0.0071
SMA transfer                =     7050.0000
Time of Flight              =        0.8182 hours

Burn A duration    185.04 s
Burn B duration    184.39 s
Thrust               0.89 N
Mass Total           6.16 kg
Mass Fuel            0.15 kg
Initial SMA       7000.00 km
Initial e            0.00
Final SMA         7099.72 km
Final e              0.00
```

12.15. Docking

Docking involves aligning the spacecraft docking adaptor with the target docking point and then controling the spacecraft so that it docks with zero relative velocity, both angular and linear, zero angular error, and zero position error. It is a six-degrees-of-freedom problem.

For this example, we will use a Soyuz spacecraft shown, with thruster locations, in Fig. 12.10. Attitude and position will be controlled simultaneously. The dynamical orbital equations are Hill's equations [2]. These are relative orbit equations linearized about a nominal circular orbit. The Hill's frame is shown in Fig. 12.9.

The dynamical equations are linear and in state-space form are:

$$\dot{s} = as + bu \tag{12.101}$$

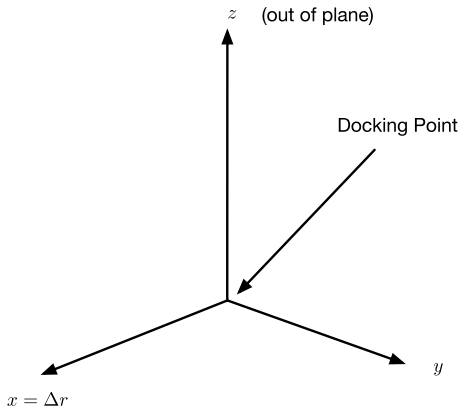


Figure 12.9 Hill's frame of reference for relative orbital motion.

where the state-transition matrix is

$$a = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3n^2 & 0 & 0 & 0 & 2n & 0 \\ 0 & 0 & 0 & -2n & 0 & 0 \\ 0 & 0 & -n^2 & 0 & 0 & 0 \end{bmatrix} \tag{12.102}$$

where n is orbit rate. The input matrix, which multiplies the control accelerations, is

$$b = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{12.103}$$

The state vector is

$$s = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{12.104}$$

where x and y are inplane. z is out-of-plane. Its dynamical equation for z is that of an undamped oscillator. x and y are coupled. y does not appear explicitly in the equations and can be large. The state equations are used to produce a full-state feedback-control law if we define the weighted control as

$$C = \frac{1}{2}\left(s^T Q s + u^T R u\right) \tag{12.105}$$

where Q is the state cost and R is the control cost. The solution, P, to the matrix Riccati equation

$$A^T P + PA - PBR^{-1}B^T P - Q = 0 \tag{12.106}$$

and the controller is

$$u = -R^{-1}B^T P s \tag{12.107}$$

or

$$u = -ks \tag{12.108}$$

where

$$k = R^{-1}B^T P \tag{12.109}$$

The force is mu, where m is the spacecraft mass. This assumes that the full state s can be measured. We also assume that the docking port is at the origin. The state measurement is done in a variety of ways. Many were demonstrated on the Swedish Space Corporation's Prisma mission [3]. Spacecraft docking with the International Space Station have multiple sensors. The sensor suite varies from spacecraft to spacecraft. Differential GPS, radar, and optical sensors are all used.

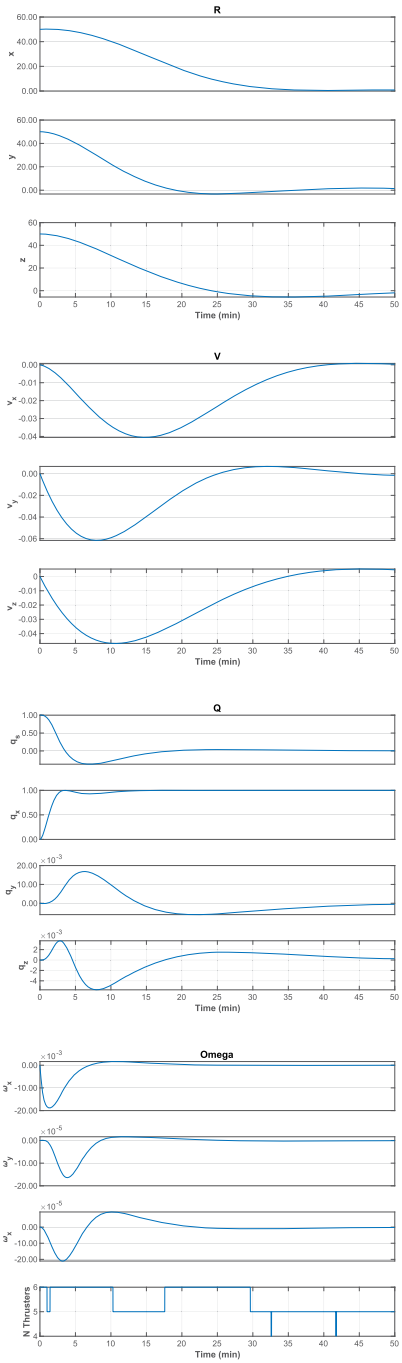
Linear programming is used to distribute forces and torques among the thrusters. Linear programming solves the problem

$$\begin{bmatrix} F \\ T \end{bmatrix} = Au \tag{12.110}$$

where u is the thruster throttle setting and A is a 6-by-n matrix of force and torque vectors, where n is the number of thrusters. The throttle setting, u, can be converted to a pulsewidth. This allows the spacecraft to easily accommodate failed thrusters. It is also easy to not use thrusters that fire towards the target spacecraft. Fig. 12.10 shows the thruster layout on a Soyuz spacecraft.

The attitude controller is a PID three-axis controller. The thrusters are pulsewidth modulated and provide translation and attitude control simultaneously. The minimum number of thrusters needed for six-degrees-of-freedom control is seven. More may be needed if saturation is considered. Example 12.13 shows the operation of the controller. In practice, the alignment operation could be done first, and then the docking.

```
syz = load('Soyuz.mat');
dT      = 0.5;
nSim    = 6000;
sMA     = 7000;
% Dynamics
n       = 2*pi/Period(sMA);
[a, b] = LinOrb( [], n );
dRHS    = syz; % Soyuz thruster data
dRHS.a = a;    % Linearized dynamics
dRHS.b = b;
% Full state feedback gain for the orbit
q       = 1e-15*eye(6);
r       = 1e-4*eye(3);
kO = QCR( a, b, q, r );
eig(a-b*kO)
% PID controller for attitude
dC      = PID3Axis;
[dC.a, dC.b, dC.c, dC.d] = PIDMIMO
      (1,1,0.004,500,0.04,dT);
dC.q_desired_state = [0;1;0;0];
% Simulate
xP = zeros(14,nSim);
x  = [50;50;50;0;0;0;1;zeros(6,1)];
A  = [dRHS.aTorque;dRHS.aForce];
err = zeros(1,nSim);
options = optimoptions('linprog','Display','none'
      );
uMax = zeros(16,1);
for k = 1:nSim
  [acc, dC] = PID3Axis( x(7:10), dC );
  torqueC = dRHS.inertia*acc;
  forceC = QForm(x(7:10),-kO*x(1:6)*dRHS.mass);
  b          = [torqueC;forceC];
  dRHS.u     = linprog(ones(1,16),[],[],A,b,zeros
      (1,16),[],[],options);
  err(k)     = Mag(b - A*dRHS.u);
  xP(:,k)    = [x;length(find(dRHS.u > 1e-8))];
  x          = RK4(@RHS,x,dT,0,dRHS);
  uMax       = max([uMax abs(dRHS.u)],[],2);
end
close %% Plotting
yL = {'x' 'y' 'z' 'v_x' 'v_y' 'v_z'...
'q_s' 'q_x' 'q_y' 'q_z' '\omega_x'  ...
'\omega_y' '\omega_x' 'N Thrusters'};
t = (0:nSim-1)*dT; k = 1:3;
TimeHistory(t,xP(k,:),yL(k),'R'); k = 4:6;
TimeHistory(t,xP(k,:),yL(k),'V'); k = 7:10;
TimeHistory(t,xP(k,:),yL(k),'Q'); k = 11:14;
TimeHistory(t,xP(k,:),yL(k),'Omega');

%% Dynamical equations
function xDot = RHS(x,~,d)
force    = d.aForce*d.u;
torque   = d.aTorque*d.u;
accel    = QTForm(x(7:10),force)/d.mass;
q        = x(7:10); omega = x(11:13);
omegaDot = d.inertia\(torque  ...
           - Cross(omega,d.inertia*omega));
qDot     = QIToBDot(q,omega);
xDot = [d.a*x(1:6)+d.b*accel;qDot;omegaDot];
end
```



Example 12.13: Docking simulation.

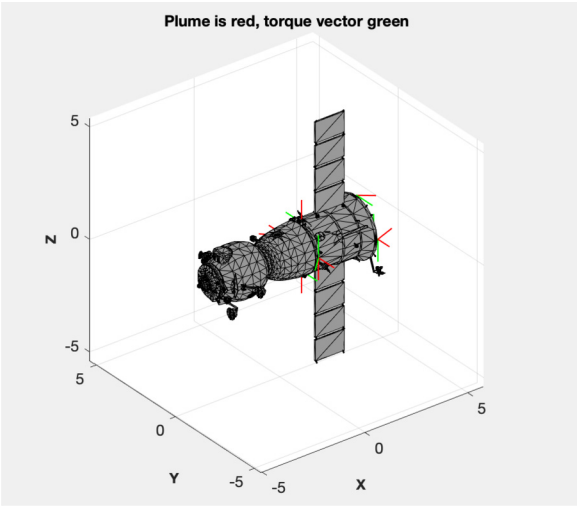


Figure 12.10 The Soyuz spacecraft with thruster locations.

12.16. Command distribution

Often, a 3-axis torque command, or a 6-axis force and torque command must be distributed to multiple actuators. One approach is to design a multioutput control algorithm with one output for each actuator. This becomes complicated when fault-tolerance and redundancy management require that the system operates with subsets of the actuators or when the number of actuators is very large (the Shuttle Orbiter has 44 thrusters). A more practical approach is to decouple the torque and force distribution from the control.

When distributing torque in this manner, it is important that the actuators have similar dynamics, and that these dynamics are accounted for in the control design. One must be careful when trying to treat a DC motor like a pulsewidth-modulated thruster!

The following sections discuss the use of techniques to distribute torque to a set of actuators. The final section discusses logic approaches.

12.16.1 The optimal torque-distribution problem

Given n actuators, all with similar dynamics as discussed above, three equality constraints, and m inequality constraints on the magnitude of the actuation, the general cost function to be minimized can be written as

$$L = C(u) + \lambda^T(T - Bu) + \mu^T u \tag{12.111}$$

where u is the control vector of order n, $C(u)$ is the scalar cost function to be applied to the control, and λ is the 3-vector of Lagrange multipliers used to adjoin the equality

constraints, T is the desired torque vector, B is the torque distribution matrix, and μ is the vector of Lagrange multipliers used to adjoin inequality constraints on the control.

Inequality constraints come in a number of forms. For a scalar actuator, one in which the torque is the product of a scalar controlled by the controller and a fixed unit vector, the effective scalar actuation is

$$u_{\text{eff}} = \begin{cases} f(u) & u_{min} \le u \le u_{max} \\ 0 & \text{otherwise} \end{cases} \tag{12.112}$$

This would apply to a fixed-orientation thruster (such as the Shuttle Orbiters Reaction Control System (RCS) thrusters, but not its Orbital Maneuvering System (OMS) thrusters) a reaction wheel, a magnetic torquer, a control-moment gyro (CMG) gimbal actuator, etc. Generally, the effective output is a nonlinear function of the input (throttle setting, voltage, pulse duration, etc.). Unfortunately, trying to minimize (12.111) with a function like (12.112) leads us to nonlinear optimization that, at present, is impractical in a real-time system.

In many cases, the actuator can be modeled as

$$u_{eff} = \begin{pmatrix} u & u \ge 0 \\ 0 & otherwise \end{pmatrix} \tag{12.113}$$

for a unidirectional actuator and

$$u_{\text{eff}} = u \tag{12.114}$$

for a bidirectional actuator such as a reaction wheel. If (12.114) is used, $\mu = 0$ and drops out of the cost function. If (12.113) is used, we still have the simple constraint on u.

The next step is to select an appropriate cost functional. For fixed-orientation thrusters, fuel is to be minimized, which means

$$C = \sum_{i=1}^{n} c_i u_i \tag{12.115}$$

For a motor energy is usually minimized

$$C = \sum_{i=1}^{n} c_i u_i^2 \tag{12.116}$$

If Eq. (12.116) is the cost functional, and (12.114) is the control, the optimal command vector can be found in a straightforward manner. Combine (12.111), (12.114), and (12.116), where (12.116) has been replaced by the equivalent vector notation, and c is a diagonal matrix whose components are the individual costs

$$L = \frac{1}{2} u^T c u + \lambda^T (T - Bu) \tag{12.117}$$

The partial derivative of L with respect to u is the vector equation

$$\frac{\partial L}{\partial u} = cu - B^T\lambda \tag{12.118}$$

Taking the derivative of (12.117) with respect to λ gives the equality constraint $T - Bu = 0$. Substituting (12.118) into the equality constraint gives the equation

$$T = Bc^{-1}B^T\lambda \tag{12.119}$$

Solving for l and replacing λ in (12.118) with the result gives

$$u = c^{-1}B^T(Bc^{-1}B^T)^{-1}T \tag{12.120}$$

which is the familiar formula for the pseudoinverse. This solution minimizes the square of the command, so it is in some sense a minimum-energy solution. It also penalizes large values of u most strongly, so it will tend to keep the actuators away from their limits. Since c and B are constant, the pseudoinverse need not be computed onboard.

12.16.2 Reaction wheels

Many Earth-pointing spacecraft use four reaction wheels in a pyramid configuration about one of the axes. That axis is usually the one about which the spacecraft rotates as it goes around its orbit. Assume that four reaction wheels are in a pyramid and each is canted θ deg from the z-axis. The control matrix is 3×4 and in this case is

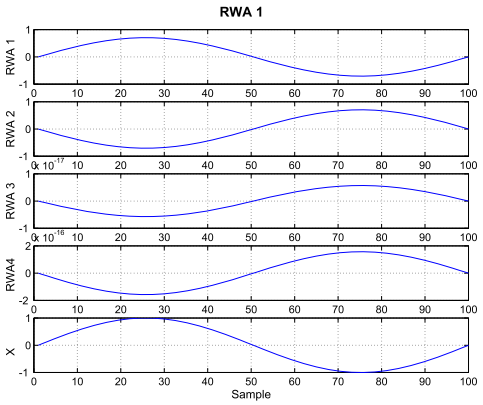
$$B = \begin{bmatrix} \sin\theta & -\sin\theta & 0 & 0 \\ 0 & 0 & \sin\theta & -\sin\theta \\ \cos\theta & \cos\theta & \cos\theta & \cos\theta \end{bmatrix} \tag{12.121}$$

If all four wheels have equal costs, c is the identity matrix and the torque distribution matrix is the 4×3 matrix

$$B^T(BB^T)^{-1} \tag{12.122}$$

All four wheels will contribute equally to any z-axis torque demand. Wheel one will respond to a positive x-axis torque demand with a positive command and wheel 2 will respond with a negative command, (12.122), which is known as the pseudoinverse function since it is the least-squares approximation to an inverse for a nonsquare matrix. Example 12.14 varies the x-axis torque sinusoidally. Note that the 1 and 2 wheels produce equal and opposite torques.

```
theta = pi/4; n = 100;
c     = cos(theta); s = sin(theta);
b     = [s -s 0 0;0 0 s -s;c c c c];
p     = pinv(b);
t     = [sin(linspace(0,2*pi,n));zeros(2,n)];
for k = 1:n
  tRWA(:,k) = p*t(:,k);
end
h = Plot2D(1:n,[tRWA;t(1,:)],'Sample',{'RWA 1' 'RWA 2' 'RWA 3' 'RWA4', 'X'})
set(h,'color',[1 1 1]);
PrintFig(0,1,1,'RWATorque')
```



Example 12.14: Reaction-wheel pyramid.

12.16.3 Linear programming

Given the problem

$$x = Au \tag{12.123}$$

$$\min(c^T u) \tag{12.124}$$

$$u \geq 0 \tag{12.125}$$

where the length of u is greater than the length of x, the solution can be found using the simplex algorithm. It is also possible to handle a magnitude constraint on u without adding additional constraint equations. Simplex is ideally suited for thrusters that are unidirectional. We could use the pseudoinverse approach if we had matched pairs of thrusters.

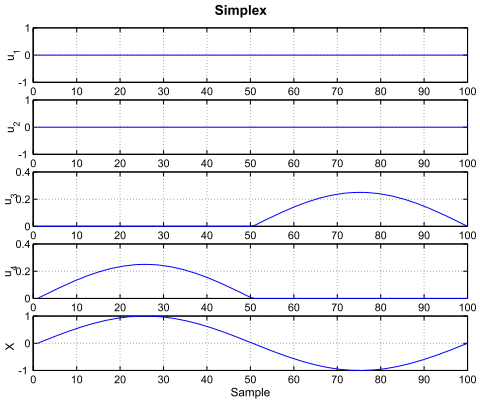
Given that x is length n and u is length m, simplex finds the subset of length n of u ($m > n$) such that

$$u = A(k,k)^{-1}x \tag{12.126}$$

where k is a subset of u of length n. Without a magnitude constraint, the number of actuators that are nonzero exactly equals the number of constraint equations. With a magnitude constraint, additional actuators will be used.

Example 12.15 is to set up North-South station-keeping control using simplex. Since we can only control two axes of torques our torque matrix is 2-by-n, not 3-by-n. Simplex picks the correct thruster and scales the thrust. Thruster scaling would have to be implemented using pulsewidth modulation.

```
n      = 100;
r      = [1 -1 0 0;0 0 1 -1;1 1 1 1];
v      = [0 0 0 0;0 0 0 0;-1 -1 -1 -1];
a      = 4*Cross(r,v);
t      = [sin(linspace(0,2*pi,n));zeros(1,n)];
for k = 1:n
  [u(:,k), f] = Simplex(ones(1,4),a(1:2,:),t(1:2,
        k),1);
end
h = Plot2D(1:n,[u;t(1,:)],'Sample',...
{'u_1' 'u_2' 'u_3' 'u_4', 'X'},...
'Simplex')
set(h,'color',[1 1 1]);
PrintFig(0,1,1,'Simplex')
```



Example 12.15: Linear programming solution for north-face thruster torque distribution.

12.17. Attitude profile design

The attitude profile of a spacecraft mission is the planned sequence of spacecraft orientations or "pointing modes" over time. The design of an attitude profile can be challenging, as it typically requires different parts of the spacecraft to simultaneously point in different directions.

For example, suppose we have a sensor to point down towards the Earth and solar panels that should point towards the Sun. The sensor is fixed to the spacecraft bus, and the solar panels may be rotated about a common axis. The attitude pointing objective, in this case, is to point the sensor boresight axis in the nadir direction and point the solar panel axis normal to the Sun vector. In this way, the solar panels may rotate about the axis to point their surface towards the Sun. Given the relative locations of the Earth and Sun with respect to the satellite, it may not be possible to completely satisfy both pointing objectives. A variety of methods may be used to determine a single orientation that provides an acceptable solution.

This section discusses one possible method for defining the attitude pointing mode as a combination of primary and secondary alignments. The primary alignment aligns a given body vector with a given target vector defined in the inertial frame. The definition of the target inertial vector depends upon the alignment type. Some of the possible alignment types include:

- Nadir pointing;
- Sun pointing;
- Orbit normal;
- Latitude-longitude pointing;
- Local vertical/local horizontal (LVLH) pointing.

The secondary alignment is achieved by rotating the spacecraft about the primary body vector, maintaining the primary alignment until the secondary body vector is as closely aligned as possible to the secondary target. The secondary target may be derived from the same menu of alignment types.

12.17.1 Alignment method

For both the primary and secondary alignments, we choose a body vector on the spacecraft to be aligned with an inertial vector. Let $\vec{b}_1$ and $\vec{b}_2$ be the body vectors for the primary and secondary alignments, respectively. Similarly, let $\vec{u}_1$ and $\vec{u}_2$ be the target inertial vectors.

The desired inertial to body quaternion for the primary alignment is found by computing the quaternion that rotates $\vec{u}_1$ to $\vec{b}_1$. This may be done with the SCT function U2Q.m.

```
>> q_1 = U2Q( u1, b1 );
```

This quaternion is defined as:

$$q_1 = \begin{bmatrix} s/2 \\ d_x/s \\ d_y/s \\ d_z/s \end{bmatrix}$$

where

$$\vec{d} = \frac{\vec{u}_1 \times \vec{b}_1}{|\vec{u}_1||\vec{b}_1|}$$

and

$$s = \sqrt{2\left(1 + \frac{\vec{u}_1 \cdot \vec{b}_1}{|\vec{u}_1||\vec{b}_1|}\right)}$$

The secondary alignment is achieved by rotating about vector $\vec{b}_1$. Let this rotation be denoted by quaternion q_{Rot}. The full ECI-to-body quaternion is therefore:

$$q_{EB} = q_1 q_{Rot}$$

where standard quaternion multiplication is used. The derivation of q_{Rot} is provided in the next section.

12.17.2 Minimizing the separation angle between vectors

Consider three unit vectors, $\vec{a}, \vec{b}, \vec{c}$, all defined in the body frame of the spacecraft. From the previous section, define $\vec{a} = \vec{b}_1$ to be the axis about which we may rotate. Define

$\vec{b} = \vec{b}_2$ as the body-fixed vector that we wish to align with vector $\vec{c}$. Finally, define $\vec{c}$ as the secondary inertial target in the body frame by rotating $\vec{u}_2$ with q_{EB}. We wish to rotate the $\vec{b}$ vector about axis $\vec{a}$ so that $\vec{b}$ is as closely aligned with $\vec{c}$ as possible.

To closely align $\vec{b}$ with $\vec{c}$, we need to minimize the separation angle between the two vectors. The separation angle α is found from the dot product definition:

$$\cos\alpha = \vec{b}^* \cdot \vec{c}^* \tag{12.127}$$

where $\vec{b}^*$ is the new value of $\vec{b}$ after rotation. Both vectors are expressed in the xyz coordinate system. The geometry is illustrated in Fig. 12.11. A new coordinate system xyz is defined with x along $\vec{a}$, $z = \vec{a} \times \vec{b}$, and y completes the right-hand system with $y = z \times x$. The $\vec{b}$ vector lies in the xy-plane. This plane is rotated through θ to place $\vec{b}^*$ in the same plane with $\vec{a}$ and $\vec{c}$.

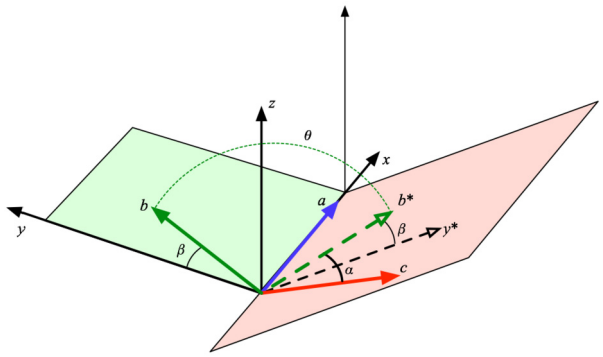


Figure 12.11 Rotation about an axis to align two vectors.

The desired vector $\vec{b}^*$ expressed in the xyz frame is:

$$\vec{b}^* = \begin{bmatrix} x^T\vec{b} \\ y^T\vec{b}\cos\theta \\ z^T\vec{b}\sin\theta \end{bmatrix} \tag{12.128}$$

The target vector $\vec{c}^*$ expressed in the xyz frame is:

$$\vec{c}^* = \begin{bmatrix} x^T\vec{c} \\ y^T\vec{c} \\ z^T\vec{c} \end{bmatrix} \tag{12.129}$$

Combining Eq. (12.128) and Eq. (12.129) into Eq. (12.127), we have:

$$\cos\alpha = (x^T\vec{b})(x^T\vec{c}) + (y^T\vec{b})(y^T\vec{c})\cos\theta + (y^T\vec{b})(z^T\vec{c})\sin\theta \tag{12.130}$$

To minimize α, we seek to maximize $\cos\alpha$. The maximum occurs when the derivative of $\cos\alpha$ with respect to θ becomes zero.

$$\frac{d(\cos\alpha)}{d\theta} = y^T\vec{b}\left(-\sin\theta y^T\vec{c} + \cos\theta z^T\vec{c}\right) = 0 \tag{12.131}$$

This leads to the simple solution:

$$\tan\theta = \frac{z^T\vec{c}}{y^T\vec{c}} \tag{12.132}$$

With the axis and angle of rotation now defined, the quaternion to perform this rotation is:

$$q_{Rot} = \begin{bmatrix} \cos(\theta/2) \\ -a_x\sin(\theta/2) \\ -a_y\sin(\theta/2) \\ -a_z\sin(\theta/2) \end{bmatrix}$$

These equations result in the following algorithm:

```
x = Unit(a);
z = Unit(Cross(x,b));
y = Unit(Cross(z,x));
m = [x';y';z'];
rotAngle = atan2(z'*ut,y'*ut);
xc = x'*b;
yc = y'*b;
ub = m'*[xc*ones(size(rotAngle)); yc*cos(rotAngle); yc*sin(rotAngle)];
c = ub(1,:).*ut(1,:)+ub(2,:).*ut(2,:)+ub(3,:).*ut(3,:);
k = find(abs(c)>1);
c(k)=sign(c(k));  % do this to prevent rounding errors from causing |c|>1
sepAngle = acos(c);
qRot = [cos(rotAngle/2);...
  sin(rotAngle/2)*axis(1);...
  sin(rotAngle/2)*axis(2);...
  sin(rotAngle/2)*axis(3)];
```

12.17.3 Computing the target-inertial vector

Let the target-inertial vector be $\vec{u}_T$. The definition of $\vec{u}_T$ is provided below for several common alignment modes. For convenient notation, let $[]_u$ represent the unit operator on a vector, returning a vector of the same direction with length 1.

12.17.3.1 Sun pointing

$$\vec{u}_T = \left[\vec{s} - \vec{r}\right]_u$$

where $\vec{r}$ is the spacecraft position in the ECI frame, and $\vec{s}$ is the position of the Sun in the ECI frame. Note that $\vec{s}$ may be found from the SCT function SunV1.m or SunV2.m by supplying the Julian date epoch.

12.17.3.2 Nadir pointing

$$\vec{u}_T = -\left[\vec{r}\right]_u$$

12.17.3.3 Latitude–longitude pointing

$$\vec{u}_T = \left[M^T \vec{g}_{EF} - \vec{r}\right]_u$$

where M is the matrix that rotates from ECI to the Earth-fixed frame, and $\vec{g}_{EF}$ is the Earth-fixed coordinates for a ground site. Given a latitude λ and longitude ℓ, the ground-site vector $\vec{g}_{EF}$ is:

$$\vec{g}_{EF} = R \begin{bmatrix} \cos\lambda \cos\ell \\ \cos\lambda \sin\ell \\ \sin\lambda \end{bmatrix}$$

Note that M may be found using the SCT function TruEarth.m.

12.17.3.4 Orbit-normal pointing

$$\vec{u}_T = \left[\vec{r} \times \vec{v}\right]_u$$

where $\vec{v}$ is the ECI velocity of the spacecraft.

12.17.3.5 LVLH pointing

$$\vec{u}_T = M^T \vec{v}$$

where M rotates from the ECI frame to a local vertical/local horizontal (LVLH) frame, and $\vec{v}$ is a unit vector defined in the LVLH frame. The rotation matrix M may be computed from the spacecraft ECI position and velocity vectors as follows:

$$M = \begin{bmatrix} x^T \\ y^T \\ z^T \end{bmatrix}$$

where $y = \left[\vec{v} \times \vec{r}\right]_u$, $z = -[r]_u$, and $x = y \times z$.

12.18. Actuator sizing

The actuators must be sized to have sufficient torque to do required attitude maneuvers and to compensate for external disturbances. Moment-exchange devices, such as reaction wheels and control-moment gyros (CMG), must have sufficient momentum-storage capability for required maneuvers and to store momentum in between momentum-dumping maneuvers if using thrusters. If you are using magnetic torquers for unloading that ARE operating all the time then the momentum-storage capacity is based on whatever the magnetic torquers cannot instantaneously remove. Sizing of magnetic-torquer systems is complicated by the changes in the planet's magnetic field as the spacecraft moves in its orbit.

12.18.1 Maneuvers

The simplest maneuver is a rotation about an axis using a bang-bang maneuver. This means the actuator is either full on or full off. Assume the desired maneuver angle is θ

$$\frac{\theta}{2} = \frac{1}{2}\frac{T}{I}\left(\frac{\tau}{2}\right)^2 \tag{12.133}$$

where T is the torque, I the inertia about that axis, and τ is the maneuver duration. The required torque is

$$T = \frac{4\theta I}{\tau^2} \tag{12.134}$$

If we are using a momentum-exchange device, the required momentum storage is

$$h_w = \frac{\tau}{2}T = \frac{2\theta I}{\tau} \tag{12.135}$$

If a momentum or reaction wheel is used the wheel speed will be

$$\omega_w = \frac{h_w}{I_w} \tag{12.136}$$

where I_w is the wheel inertia. If a CMG is used the CMG angle is

$$\theta = \sin^{-1}\left(\frac{h_w}{h_{CMG}}\right) \tag{12.137}$$

where h_{CMG} is the momentum stored in the CMG.

The result will be slightly different for different controllers. Assume a PD controller with $\zeta = 1$. The second-order equation is

$$I\left(\ddot{\theta} + 2\omega_n\dot{\theta} + \omega_n^2\theta\right) = T \tag{12.138}$$

where ω_n is the natural frequency of the controller. The forward gain is $I\omega_n^2$. The solution is from the characteristic equation

$$s^2 + 2\omega_n s + \omega_n^2 = 0 \tag{12.139}$$

The solution for repeated roots is

$$\theta(t) = c_1 e^{-\omega t} + c_2 t e^{-\omega t} \tag{12.140}$$

If $\theta(0) = \theta_0$ and $\dot{\theta(0)} = 0$. The coefficients are

$$c_1 = \theta_0 \tag{12.141}$$

$$c_2 = \omega_n \theta_0 \tag{12.142}$$

$$\theta = \theta_0 e^{-\omega_n t}(1 + \omega_n t) \tag{12.143}$$

The control torque is

$$T = -I\left(2\omega_n \dot{\theta} + \omega_n^2 \theta\right) \tag{12.144}$$

or

$$T = \omega_n^2 I \theta_0 e^{-\omega_n t}(1 - \omega_n t) \tag{12.145}$$

The peak torque is

$$T = \omega_n^2 I \theta_0 \tag{12.146}$$

This can be compared with a bang-bang maneuver by selecting an acceptable final angle, θ_f, and computing ω_n numerically from the equation

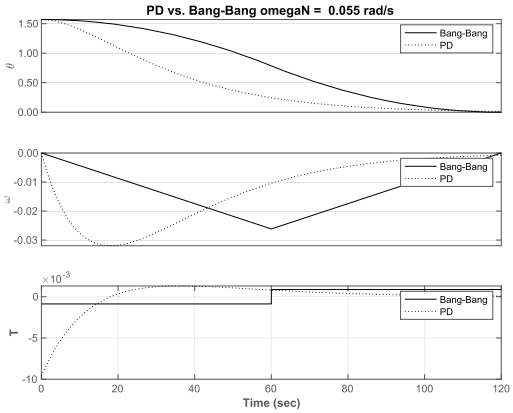
$$\frac{\theta_f}{\theta_0} = e^{-\omega_n \tau}(1 + \omega_n \tau) \tag{12.147}$$

Example 12.16 compares a bang-bang maneuver with a PD maneuver. The PD requires much larger torque at the start. The PD controller reaches a higher angular velocity that will result in more momentum stored in the momentum-exchange device if momentum-exchange devices are used. The peak torque is also higher than with the bang-bang maneuver.

12.18.2 Disturbances

Estimating the size of momentum-exchange devices for disturbance accommodation can be done using a Fourier-series representation of the disturbance torque. For exam-

```
%% PD vs bang bang maneuver

inr          = 2;
theta0       = pi/2;
tau          = 120;
torqueBBMax  = 4*theta0*inr/tau^2;

thetaF       = 0.01*theta0;
omegaN       = FindOmegaN(thetaF/theta0,tau);
torquePDMax  = (theta0*inr)*omegaN^2;

n            = 100000;
dT           = tau/n;
xP1          = zeros(3,n+1);
xP2          = zeros(3,n+1);
x            = [theta0;0];
torque       = -torqueBBMax;
for k = 1:n
  xP1(:,k) = [x;torque];
  if( k < n/2)
    torque  = -torqueBBMax;
  else
    torque  =  torqueBBMax;
  end
  x        = RK4(@RHS,x,dT,0,torque,inr);
end
xP1(:,k+1) = [x;torque];

x = [theta0;0];
for k = 1:n
  torque   = -inr*omegaN*(2*x(2) + omegaN*x(1));
  xP2(:,k) = [x;torque];
  x        = RK4(@RHS,x,dT,0,torque,inr);
end
xP2(:,k+1) = [x;torque];

t  = (0:n)*dT;
yL = {'\theta', '\omega', 'T'};
lg = {'Bang-Bang' 'PD'};
kP = {[1 4] [2 5] [3 6]};
s  = sprintf('PD vs. Bang-Bang omegaN = %6.3f rad/s',omegaN);
TimeHistory(t,[xP1;xP2],yL,s,kP,{lg lg lg});

function xDot = RHS(x,~,t,inr)

xDot = [x(2);t/inr];

end

function omegaN = FindOmegaN(r,tau)

myfun = @(x,tau,r) r - exp(-x*tau)*(1+x*tau);
omegaN = fzero( @(x) myfun( x, tau, r), [0, 1] );
end
```



Example 12.16: Torque comparison for a 90-degree rotational maneuver.

ple, a sine/cosine series is

$$\frac{dh_w}{dt} = a_0 + \sum_{k=1}^{N} \left(a_k \sin(k\omega_o t) + b_k \cos(k\omega_o t) \right) \tag{12.148}$$

where ω_o is the orbit rate. This is for a planet pointing satellite. The momentum stored is

$$h_w = h_w(0) + a_0 t - \frac{1}{\omega_o} \sum_{k=1}^{N} \frac{1}{k} \left(a_k \cos(k\omega_o t) - b_k \sin(k\omega_o t) \right) \tag{12.149}$$

The momentum wheel experiences a steady increase due to a_0. The harmonics will cause periodic peaks. This means unloading will need to be done earlier than what would be calculated by just considering $a_0 t$. Maneuver planning must include any momentum already stored in the momentum-exchange device, or any wheel-speed biasing that may have been done.

The magnitude of the disturbances is typically on the order of nano-N. For microthrusters that operated continuously, the thruster must produce a torque greater than the disturbance. For larger thrusters, such as typical hydrazine thrusters, it determines the minimum pulsewidth and the control period. Assume that the disturbance is the constant a_0 given above. If the thruster torque is T then the minimum pulsewidth must be

$$\tau_m = \frac{a_0}{T}\tau \tag{12.150}$$

where τ is the pulsing period. The achievable τ_m will be determined by the valve dynamics.

The torque from a magnetic torquer is

$$T = M^{\times} B \tag{12.151}$$

where M is the dipole moment and B is the magnetic field in the body frame. A magnetic control system cannot produce a torque along the direction of B. Consequently, it is not possible to always instantaneously cancel a disturbance torque in all three axes. The sizing of the magnetic torquers is very dependent on the control algorithm. It is also very dependent on the knowledge of the planet's magnetic field. The field is dependent on the spacecraft's orbital location above the planet, the angle between the spacecraft position vector and the Sun vector, and the solar activity. The magnetic field of a planet also changes slowly with time. This complicates the prediction of the field. A magnetometer could measure the field, and the previous orbits measurements used to improve future predictions.

Sizing of the torquers can be done by taking the total momentum growth over and orbit, h, and computing the minimum dipole needed to cancel it. That is

$$h = \int_0^{\tau} M^{\times} B dt \tag{12.152}$$

where τ is the orbital period. This can be done by minimizing this cost functional over one or more orbits

$$J = (T_d - M^{\times} B)^T (T_d - M^{\times} B) + M^T M \tag{12.153}$$

The first term minimizes the quadratic torque error. The second minimizes the magnitude of the dipole. The solution is

$$S = \int_0^\tau B^\times B^\times dt \tag{12.154}$$

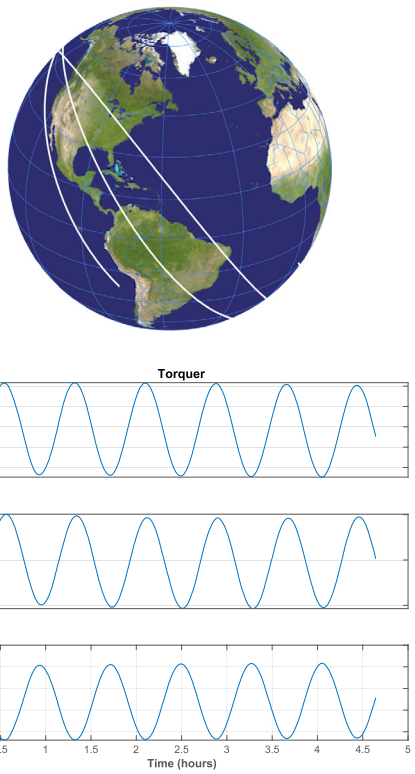
$$\lambda = S^{-1} h \tag{12.155}$$

$$M(t) = \lambda^\times B(t) \tag{12.156}$$

This weights the torque error and control-dipole magnitude the same.

Example 12.17 computes the optimal magnetic dipole over three ISS orbits assuming a constant amount of momentum that needs to be removed. This is equivalent to a constant disturbance torque. The example algorithm works in the inertial frame. The integrated torque exactly matches the desired value.

```
%% Compute the optimal magnetic dipole

tD          = [1;2;3]*1e-6;
n           = 1000;
[el, jD0]   = ISSOrbit;
p           = Period(el(1));
tau         = 3*p;
t           = linspace(0,tau,n);
r           = RVOrbGen(el,t);

PlotOrbit(r,t,jD0)
jD          = jD0 + t/86400;
b           = BDipole(r,jD);
s           = zeros(3,3);
for k = 1:n
   s = s + SkewSq(b(:,k));
end

h       = tD*tau;
lambda = s\h;

m  = zeros(3,n);
tC = zeros(3,1);
for k = 1:n
   m(:,k) = cross(lambda,b(:,k));
   tC     = tC + cross(m(:,k),b(:,k));
end
tC/tau
TimeHistory(t,m,{'M_x' 'M_y' 'M_z'},'Torquer')
```



Example 12.17: The optimal magnetic dipole to cancel a constant disturbance torque.

References

[1] M. Bagheri, M. Kabganian, R. Nadafi, Three-axis attitude control design for a spacecraft based on Lyapunov stability criteria, Scientia Iranica 20 (4) (2013) 1302–1309.
[2] D.A. Vallado, Fundamentals of Astrodynamics and Applications, 2nd ed., McGraw-Hill, 1997.
[3] SSC, PRISMA, http://www.ssc.se/?id=7611.

CHAPTER 13

Momentum control

13.1. Space story

The first-shift attitude control engineers had left the momentum-unloading system running at shift change. This system used thrusters to remove pitch momentum from the momentum wheel. The momentum-unloading software was not smart enough to stop when the wheel was unloaded. The operator was on a break and it continued to fire thrusters that knocked the bias-momentum axis off orbit normal and induced large nutation. I got the operator back on the console and we began a series of thruster firings to fix the offset and damp nutation. This was done by phasing the thruster firings with the gyro measurements. As we were getting it under control, the customer called and asked if everything was alright. We responded, "Everything is just fine."

13.2. Introduction

Momentum unloading is used on spacecraft that have momentum-exchange devices for attitude control. Momentum-exchange devices include reaction wheels and a control-moment gyro. Momentum-exchange devices only exchange momentum between the device and the spacecraft, they can never remove momentum from the spacecraft. If there is a secular, that is an inertial-frame fixed torque, the momentum-exchange devices will eventually run out of momentum storage and need to be unloaded. To unload the momentum, you need an external torque. This can be done with magnetic torquers if your planet has a magnetic field, thrusters, solar pressure, or aerodynamic forces.

In this chapter, we break the problem into momentum growth, the control algorithm, and then torque generation.

13.3. Momentum growth

It is easiest to look at the inertial frame dynamical equations that are simply

$$T = \dot{H} \tag{13.1}$$

where T is the disturbance torque in the inertial frame and H is the inertial spacecraft momentum. Assume that the torque is a combination of a steady torque and a sinusoidal torque

$$T = T_0 + T_s \sin \omega_o t \tag{13.2}$$

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00024-3

where t is time and ω_o is the orbital rate of the satellite. This is typical for a planet-pointing satellite. If the initial momentum is zero we get

$$H = T_0 t + \frac{T_s}{\omega_o}(1 - \cos \omega_o t) \tag{13.3}$$

The momentum will grow linearly with time and oscillate at the orbit rate. If the momentum-exchange devices have sufficient momentum-storage capability, the orbit-rate momentum will be absorbed and released over the orbit. The linear momentum growth will need to be unloaded.

13.4. Control algorithms

One approach is to use a Proportional-Integral (PI) controller of the form

$$T = -k\left(H + \frac{1}{\tau_I}\int H\right) \tag{13.4}$$

The first term generates a torquer proportional to the momentum. The second produces a torque proportional to the integral of the momentum. This drives the error to zero, without any offset. We write the equations of motion with the control ω for the control is set to be ten times orbit rate.

$$T_d = \dot{H} + k\left(H + \frac{1}{\tau_I}\int H\right) \tag{13.5}$$

where T_d is the disturbance torque. Integrate once

$$H_d = \ddot{H} + k\dot{H} + \frac{k}{\tau_I}H \tag{13.6}$$

which is a second-order damped oscillator. The controller gains are related to the damping ratio and the undamped natural frequency by

$$k = 2\zeta\omega \tag{13.7}$$

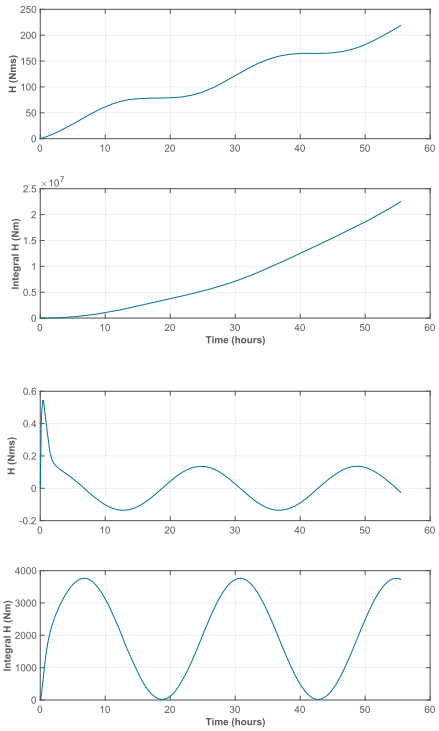
$$\tau_i = \frac{2\zeta}{\omega} \tag{13.8}$$

Typically, you want your controller to be faster than the disturbance frequency. The damping ratio, ζ should be greater than 0.7071, the critical damping ratio. One is often chosen for ζ since overshoot is undesirable.

Example 13.1 shows the results with and without control for a single axis. The control system is built into the right-hand-side dynamical model and can be turned on and off via the data structure. When the control is off, the momentum grows linearly,

as expected with an oscillatory component. When the control is on, the integral term drives the offset to zero.

```
cOn = true; % Control is on if true
x   = [0;0]; % [H;integral H]
n   = 2000; % Number of steps
dT  = 100; % Seconds
omega = 2*pi/8640;
zeta  = 1;

xP  = zeros(2,n); % Store plot data
d   = struct('k',2*zeta*omega,...
             'tau',2*zeta/omega,...
             'tD',0.001,'tS',0.0001,...
             'omega',2*pi/86400);
if( ~cOn ) % Set the forward gain
  d.k = 0;
end
t  = (0:n-1)*dT;
for k = 1:n
  xP(:,k) = x;
  x = RK4(@RHSMomentum,x,dT,t(k),d);
end
yL = {'H (Nms)','Integral H (Nm)'};
TimeHistory(t,xP,yL,'Momentum');

%% Right hand side
function xDot = RHSMomentum(x,t,d)
f    = d.tD + d.tD*sin(d.omega*t)...
     - d.k*(x(1) + x(2)/d.tau);
xDot = [f;x(1)];

end
```



Example 13.1: Momentum control. The right-hand side for the momentum dynamical equations is a function at the bottom of the listing.

13.5. Control-torque generation

13.5.1 Thruster control

13.5.1.1 Direct

If we are using thrusters for momentum unloading we just transform the inertial torque demand into the body frame

$$T_B = AT_{ECI} \tag{13.9}$$

where A is a transformation matrix from the inertial to the body frame. This can be found in your attitude quaternion. It assumes you have three-axis information. The torque generated by the thrusters is

$$T_B = \sum_i (r_i - c)^{\times} F_i \tag{13.10}$$

where F is the thruster force vector, r_i is the location of the ith thruster with respect to the reference point and c is the location of the center-of-mass with respect to the same reference frame. The control law given in the last section is linear and proportional. This means the torque demand, and therefore the force demand will go from zero to the maximum force the thruster can provide. You may note that one thruster cannot provide any three-axis torque because $(r_i - c)^\times$ for just one thruster is not invertible. A sum of thrusters will be as long as they have moment arms so that three-axis torques are possible.

Small thrusters are rarely throttleable. Therefore, we approximate a linear thruster by using pulsewidth modulation. Pulsewidth modulation is turning on a thruster for only a portion of the control period. If the thruster has a force F_{max} and we want the average force over the control period, τ to be F we just turn the thruster on for part of the control period

$$\tau_i = \frac{F_i}{F_{max}}\tau \tag{13.11}$$

where τ_i will have a minimum. You do not need to make τ the same length as your attitude control period. Since momentum unloading is usually slow you can make it much lower than your control period, which gives your momentum-unloading system a much higher resolution.

13.5.1.2 Off-pulsing

Off-pulsing is a method of getting attitude control from thrusters that are normally on. Suppose you have thrusters on the four corners of your spacecraft that are turned on to get thrust in that body's direction. For example, in a geosynchronous satellite, you might use four East-faced thrusters for an East-West station-keeping maneuver. If the thrusters are oriented and located so that you do not get a net torque, the attitude is undisturbed. The situation is shown in Fig. 13.1. Off-pulsing is exactly like onpulsing. Instead of turning on a thruster for the pulsewidth, you delay turning on a thruster for the computed pulsewidth duration. This puts the torque generation closest to the sensor measurement. If you instead, shut the pulse off early, you would have almost a pulse-period delay. The sign of the torque is flipped for the thruster selection. The thruster that has the longest pulsewidth produces less torque than the opposite thruster.

13.5.2 Magnetic control

13.5.2.1 Magnetic field

If the planet about which you are orbiting has a magnetic field, then you can use magnetic torquers for momentum control. Planets and moons with significant magnetic fields are

- Earth;

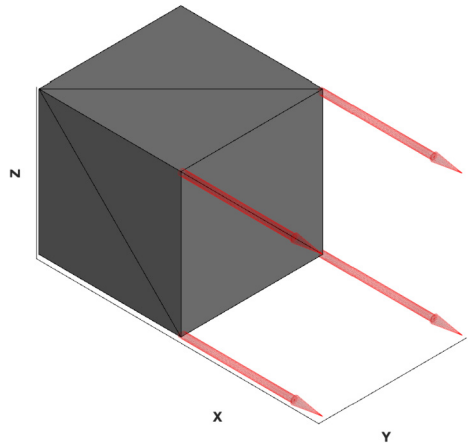


Figure 13.1 Four thrusters on for a velocity-change maneuver. The arrows show the plume direction.

- Jupiter;
- Saturn;
- Uranus;
- Neptune;
- Ganymede;
- Europa.

The simplest approximation to a planet's magnetic field is a magnetic dipole oriented along the magnetic axis, which is not necessarily the rotation axis. Example 13.2 shows a dipole model of the Earth in three different orbits, equatorial, ISS, and polar. The field is given in the ECI frame. In the polar and equatorial orbits, one component of the magnetic field is constant. This model is a dipole. The magnetic field is also affected by the Sun and that may have to be considered, especially for high orbits.

13.5.2.2 Instantaneous control

We transform the inertia frame torque demand into the body frame

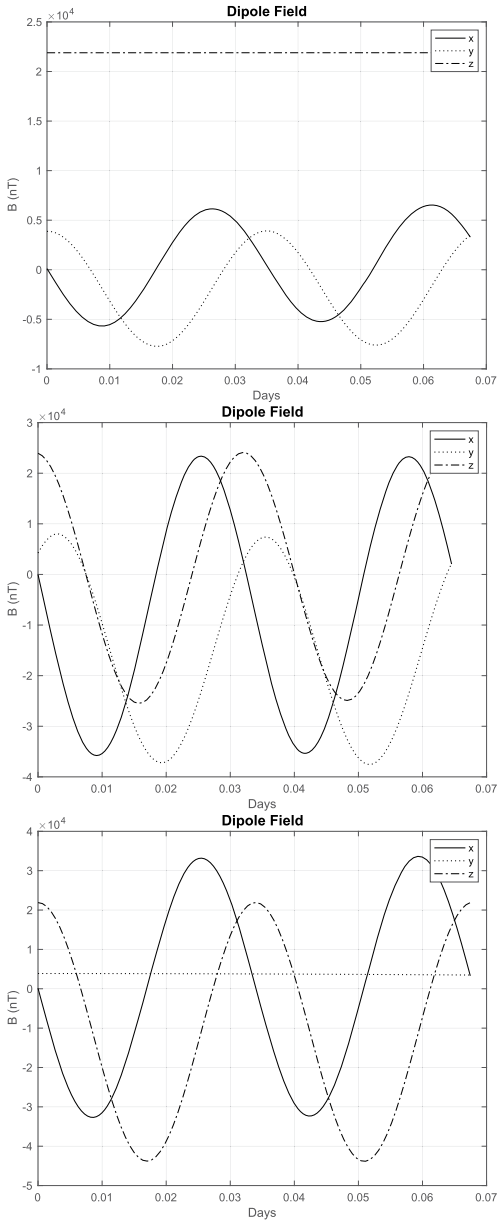
$$T_B = A^T T_{\text{ECI}} \tag{13.12}$$

where A is the transformation matrix from the body to the ECI frame. A is known from the measured attitude quaternion. The torque is produced by magnetic torquers

$$T_B = M^{\times} B \tag{13.13}$$

where M is the magnetic torquer dipole and B is the magnetic field measured by a magnetometer or from an onboard magnetic-field model if we know the orbital position

```
%% ISS
[el,jD0] = ISSOrbit;
% Returns t for an orbit period
[r,~,t]  = RVOrbGen(el);
% Julian date is for the rotation of the Earth
jD = jD0 + t/86400;
BDipole(r,jD)

%% Polar
% Elements are [semi-major axis, inclination ...]
[r,~,t]  = RVOrbGen([7000 pi/2 0 0 0 0]);
jD = jD0 + t/86400;
BDipole(r,jD)

%% Equatorial
[r,~,t]  = RVOrbGen([7000 0 0 0 0 0]);
jD = jD0 + t/86400;
BDipole(r,jD)
```



Example 13.2: The Earth's magnetic field in equatorial, ISS, and polar orbits. The equatorial and polar orbits are at radii of 7000 km.

accurately. For some orbits, like geosynchronous, the magnetic field is fairly constant so measurements are not required. B is in the body frame. Flip the order of M and B using a vector identity

$$T_B = -B^{\times} M \tag{13.14}$$

Since the first term is skew-symmetric, it cannot be inverted to find M. Instead, we do a least-squares fit.

$$M = \frac{B^{\times} T_B}{|B|^2} \tag{13.15}$$

The simplest way to implement the unloading is to pulsewidth modulate the magnetic torquers, just like with thrusters. A magnetic torquer can then be always maximum on or off. It is controlled by a switch. If you wanted a variable dipole, you would need to add a linear amplifier, which would waste energy. The magnetic-torquer pulse period is selected that is n times as long as the control period because unloading can be a slower process than control. Let $M_{\max}$ be the saturated value of the dipole. The fraction of the time on is

$$f = \frac{|M|}{M_{\max}} \tag{13.16}$$

The torquer coil is on for n_{on} cycles

$$n_{\text{on}} = \text{round}(fn) \tag{13.17}$$

The polarity is the sign of M.

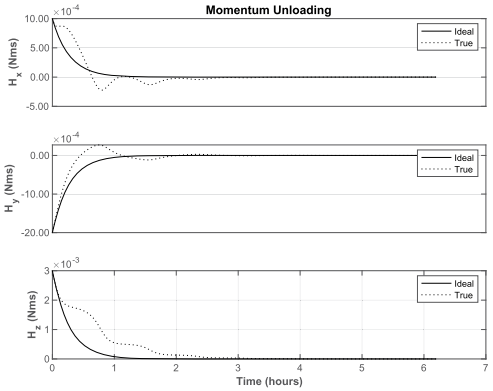
Example 13.3 shows the transient response of a magnetic-torquer system. It is removing momentum stored at the beginning of the simulation. Only proportional control is used. The simulation compares the magnetic-torquer control with an ideal proportional controller. The spacecraft is in an ISS orbit. The dipole moment is determined by the forward gain, k.

The ideal torque reduces the momentum smoothly. The magnetic control eventually removes all of the momentum but increases it on the x-axis at the beginning.

13.5.2.3 Individual torquer control

Three torquers are not needed to unload momentum. Another approach is to find the component of the torquer demand along a particular magnetic-torquer torque direction, set the dipole to the fraction of the dipole moment that produces that torque, subtract

```
secInDay = 86400;
n        = 1000;
[el,jD0] = ISSOrbit;
jDStart  = Date2JD;
nOrbits  = 4;
gain     = 0.001; % Unloading gain

%% Orbit
p        = Period(el(1));
t        = linspace(0,nOrbits*p,n);
jD       = jD0 + t/secInDay;
[r,v]    = RVOrbGen(el,t);

%% Magnetic field
b        = BDipole(r,jD);

% Set up the plotting arrays
h0        = [1;-2;3]*1e-3;
hECI      = zeros(3,n);

% Orients the spacecraft with local
% vertical and local horizontal
q         = QLVLH(r,v);
b         = QForm(q,b);
h         = QForm(q(:,1),h0);
hECI(:,1) = h0;
hECIM     = hECI;
mDipole   = zeros(3,n);
dT        = t(2); % Time step

%% Simulate unloading a constant momentum
for k = 2:n
  sB                  = Skew(b(:,k-1));
  tMM                 = gain*hECIM(:,k-1);
  mDipole(:,k-1)      = pinv(sB*sB)*sB*tMM;
  tIdeal              = -gain*hECI(:,k-1);
  hECI(:,k)           = hECI(:,k-1) + dT*tIdeal;
  mB                  = Cross(mDipole(:,k-1),b(:,
        k-1));
  hECIM(:,k)          = hECIM(:,k-1) + dT*mB;
end

TimeHistory(t,[hECI;hECIM],...
  {'H_x (Nms)' 'H_y (Nms)' 'H_z (Nms)'},...
  'Momentum Unloading', {[1 4] [2 5] [3 6]},...
 {{'Ideal' 'True'} {'Ideal' 'True'} {'Ideal' '
       True'}});
```



Example 13.3: Proportional-control momentum unloading. Shows ideal proportional control and control with magnetic torquers.

from the torque demand, and then cycle through the rest of the torquers.

$$T_M = M^{\times} B \tag{13.18}$$

$$u_M = \frac{T_M}{|T_M|} \tag{13.19}$$

$$T_C = (T_B^T u_M) u_M \tag{13.20}$$

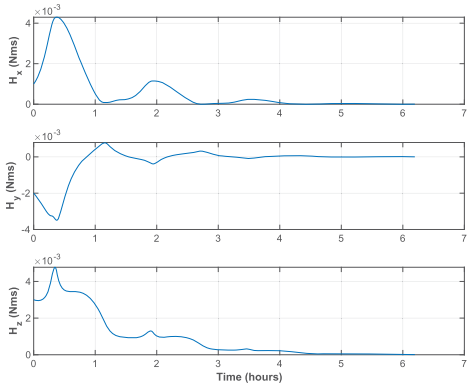
$$m = \frac{|T_C|}{|T_M|} \tag{13.21}$$

$$T_B = T_B - T)C \tag{13.22}$$

where m is the dipole fraction. You repeat this loop for each torquer.

Example 13.4 shows the transient response of a magnetic-torquer system with only two torquers. It uses the same control algorithm as the previous example.

```
% The torquers
m          = [1 0;0 1;0 0];
nM         = size(m,2);

secInDay = 86400;
n          = 1000;
[el,jD0] = ISSOrbit;
jDStart  = Date2JD;
nOrbits  = 4;
gain       = 0.001; % Unloading gain

%% Orbit
p          = Period(el(1));
t          = linspace(0,nOrbits*p,n);
jD         = jD0 + t/secInDay;
[r,v]      = RVOrbGen(el,t);

%% Magnetic field
b          = BDipole(r,jD);

% Initial momentum
h0         = [1;-2;3]*1e-3;

% Set up the plotting array
hECI       = zeros(3,n);

% Orients the spacecraft with local
% vertical and local horizontal
q          = QLVLH(r,v);
b          = QForm(q,b);
h          = QForm(q(:,1),h0);
hECI(:,1) = h0;
mDipole    = zeros(3,n);
dT         = t(2); % Time step

%% Simulate unloading a constant momentum
for k = 2:n
  tD     = gain*hECI(:,k-1);
  mD   = zeros(3,1);

  % Loop through the torquers
  for j = 1:nM
    tM     = Cross(m(:,j),b(:,k-1));
    uTM    = Unit(tM);
    tCK    = (tD'*uTM)*uTM;
    mD     = mD + Mag(tCK)*m(:,j)/Mag(tM);
    tD     = tD - tCK;
  end
  mB               = Cross(mD,b(:,k-1));
  hECI(:,k)       = hECI(:,k-1) + dT*mB;
end

TimeHistory(t,hECI,{'H_x (Nms)' 'H_y (Nms)' 'H_z
      (Nms)'},...
  'Momentum Unloading');
```



Example 13.4: Individual torquer-momentum unloading. Shows ideal proportional control and control with two magnetic torquers.

Ideally, the momentum-control torque reduces the momentum smoothly. In this example, the magnetic control eventually removes all of the momentum but does increase the momentum on the x-axis at the beginning of the momentum-unloading operation.

13.5.2.4 Average control

While a magnetic torquer cannot produce a specific three-axis torque at a given time, it may be able to produce an average three-axis torque that meets the demand if the magnetic field varies sufficiently over time.

The torque produced by a torquer of dipole m is

$$T = M^{\times} B \tag{13.23}$$

where the skew-symmetric matrix for M is

$$M^{\times} = \begin{bmatrix} 0 & -M_z & M_y \\ M_z & 0 & -M_z \\ -M_y & M_z & 0 \end{bmatrix} \tag{13.24}$$

The above matrix cannot be inverted; consequently, it is not possible to realize a three-axis torque demand with magnetic torquers. However, if B changes it will be possible on average to satisfy an average three-axis demand. Assume that we want to drive the momentum vector h to zero over an interval T. Write the Hamiltonian, which is the cost you want to minimize. This applies a cost to the quadratic of the dipole and adjoins the right-hand-side of the dynamical equations

$$\mathcal{H} = \frac{1}{2} M^T M - \lambda^T B^{\times} M \tag{13.25}$$

where λ are the Lagrange multipliers that must be found as part of the solution. The control law is found by taking the derivative of $\mathcal{H}$ with respect to m and setting the result to zero

$$M = -B^{\times} \lambda \tag{13.26}$$

where λ is a constant vector. Therefore, substituting the above into the constraint equation gives

$$h = \int_0^T B^{\times} M dt \tag{13.27}$$

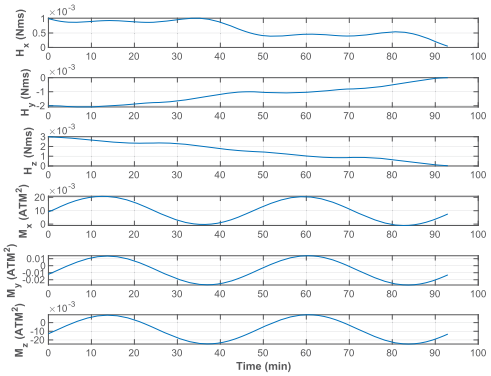
$$h = -\left[\int_0^T B^{\times} B^{\times} dt \right] \lambda \tag{13.28}$$

$$\lambda = \left[\int_0^T B^{\times} B^{\times} dt \right]^{-1} h \tag{13.29}$$

If the integral is invertible, it is possible to drive h to zero over the interval T. It is necessary to know B over the time interval T, which requires an onboard ephemeris or recording magnetometer readings from the previous orbit. Example 13.5 shows average

control. The momentum is driven to zero over one orbit. The starting momentum is the same as in Example 13.3. That controller reduces the momentum to near zero in one orbit but takes several to completely zero out the momentum.

```
h0          = [1;-2;3]*1e-3;

%% ISS
[el,jD0]          = ISSOrbit;

% Returns t for an orbit period
[r,~,t]     = RVOrbGen(el);
n           = length(t);

% Julian date is for the rotation of the Earth
jD          = jD0 + t/86400;
b           = BDipole(r,jD);
f           = zeros(3,3);
dT          = t(2);
for k = 1:n
   s = Skew(b(:,k));
   f      = f + s*s;
end
lambda = dT*f\h0;
h       = zeros(3,n);
h(:,1) = h0;
m       = Cross(lambda,b);
for k = 1:n-1
   h(:,k+1) = h(:,k) - dT*Cross(m(:,k),b(:,k));
end
yL = {'H_x␣(Nms)' 'H_y␣(Nms)' 'H_z␣(Nms)' ...
      'M_x␣(ATM^2)' 'M_y␣(ATM^2)' 'M_z␣(ATM^2)'
                 };
TimeHistory(t,[h;m],yL,'Momentum')
```



Example 13.5: Average control.

13.5.3 Solar and aerodynamic pressure

13.5.3.1 Introduction

Solar and aerodynamic forces put torques on the spacecraft. They are major sources of disturbances and momentum growth. They can also be used as actuators. Normally, one would not add surfaces for control but it is possible to take advantage of either type of force for momentum management. Two examples will be given. One is establishing a torque-equilibrium attitude using aerodynamic forces. The other is using the solar wings on a satellite for momentum control.

13.5.3.2 Torque-equilibrium attitude

If your spacecraft has the flexibility to change its orientation, it may be possible to fly it in a torque-equilibrium attitude to balance out the disturbance torques. The most common situation is to balance gravity-gradient and aerodynamic torques in low-Earth orbit about the Earth-pointing axis, which is the LVLH $+z$ or yaw axis. The gravity-gradient torque in the body frame is

$$t_g = \mu \frac{r_b^{\times} I r_b}{|r_b|^5} \tag{13.30}$$

where I is the inertia matrix, μ is the gravitational constant, and r_b is the position vector of the spacecraft in the body frame. r_b is a function of the spacecraft attitude. The aerodynamic torque is the sum of the aerodynamic forces on all of the elements of the spacecraft that are not shadowed for the gas flow around the spacecraft. The force on each element is opposite the velocity vector. As we showed in the disturbances chapter, the actual force vector may have a component transverse to the normal, but that would not change the analysis. Defined the dynamic pressure as

$$p = \frac{1}{2}\rho|v|^2 \tag{13.31}$$

where ρ is the atmospheric density and v is the velocity. The aerodynamic torque is

$$t_a = -p\sum_i C_{D_i}(n_i^T u_b)a_i(r_i - c)^\times u_b \tag{13.32}$$

where C_{D_i} is the drag coefficient of plate i, a_i is the area of plate i, n_i is the surface normal of the plate in the body frame, c is the location of the center-of-mass, and u_b is

$$u_b = \frac{v_b}{|v_b|} \tag{13.33}$$

The geometry is shown in Fig. 13.2. We want to find an attitude where

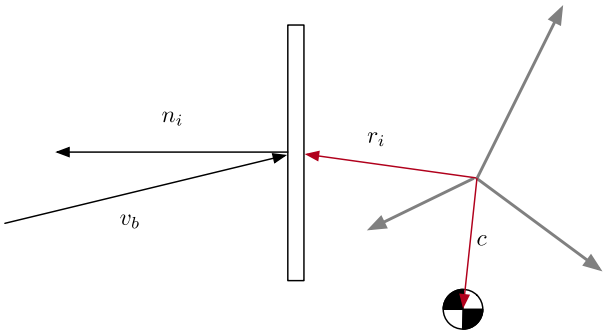


Figure 13.2 Aerodynamics geometry.

$$t_a + t_g = 0 \tag{13.34}$$

We can explore this with a simple case. Assume our spacecraft is a single plate with its normal along the velocity vector, which is the y-axis. The position vector is along the body z-axis. The center-of-mass is at zero and the plate is offset along the x-axis. The geometry is shown in Fig. 13.3. When the spacecraft is aligned with the LVLH frame there is no aerodynamic force on the spacecraft. Eq. (13.34) becomes

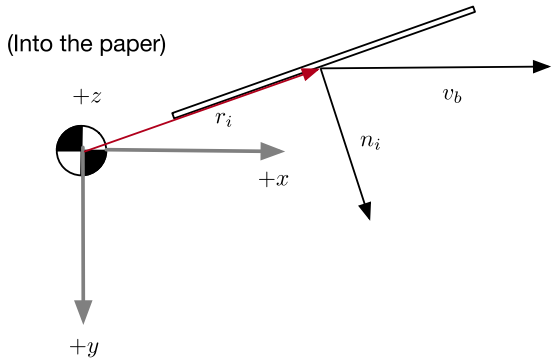


Figure 13.3 Simplified aerodynamics geometry.

$$t_a + \omega_o^2 u^{\times} I u = 0 \tag{13.35}$$

where u is the unit orbit vector in the body frame and x is the x coordinate of the plate moment arm. The gravity-gradient term was simplified using the orbit rate

$$\omega_0^2 = \frac{\mu}{|r|^2} \tag{13.36}$$

When the position vector is aligned with the body z-axis the gravity-gradient torque is

$$t_g = \omega_o^2 \begin{bmatrix} -I_{yz} \\ I_{xz} \\ 0 \end{bmatrix} \tag{13.37}$$

We can compensate for this disturbance by tilting the spacecraft. If the angles are small, the transformation matrix from LVLH to the body is

$$A = \begin{bmatrix} 1 & \gamma & -\beta \\ -\gamma & 1 & \alpha \\ \beta & -\alpha & 1 \end{bmatrix} \tag{13.38}$$

The angles and signs can be found by inspection by looking at Fig. 13.4. The unit position and velocity vectors in the body frame are

$$u = \begin{bmatrix} -\beta \\ \alpha \\ 1 \end{bmatrix} u_b = \begin{bmatrix} 1 \\ -\gamma \\ \beta \end{bmatrix} \tag{13.39}$$

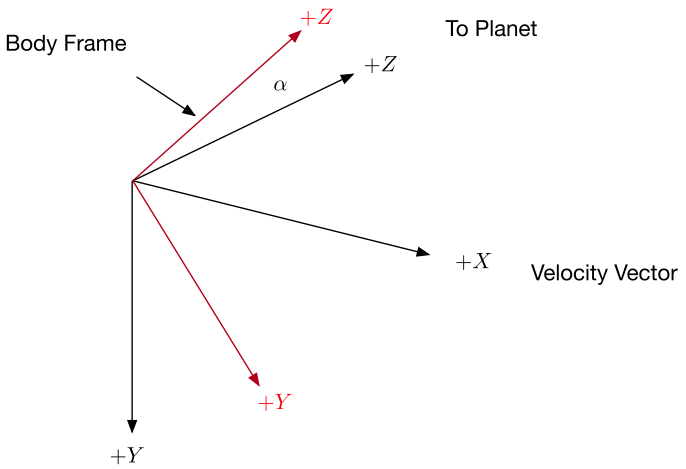


Figure 13.4 Coordinate transformations from LVLH to body with small angles. This diagram shows the roll angle, α. The body axes are in red.

The unit vectors and the transformation matrix are approximations. The vectors are not exactly unit vectors and the transformation matrix is not exactly orthonormal. If the angles are small, they are sufficiently accurate for a linear approximation.

The linearized aerodynamic torque is

$$t_a = -pC_D a \left(\begin{bmatrix} 0 \\ n_x r_z \\ -n_x r_y \end{bmatrix} + \begin{bmatrix} 0 & r_y n_x & r_z n_x \\ 0 & r_z n_z - r_x n_x & -r_z n_y \\ 0 & -r_y n_z & r_y n_y - r_x n_x \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \right) \tag{13.40}$$

The linearized gravity-gradient torque is

$$t_g = -\omega_o^2 \left(\begin{bmatrix} -I_{yz} \\ I_{xz} \\ 0 \end{bmatrix} + \begin{bmatrix} I_{zz} - I_{yy} & I_{xy} & 0 \\ I_{xy} & I_{zz} - I_{xx} & 0 \\ -I_{xz} & -I_{yz} & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \right) \tag{13.41}$$

The angle vector can be solved from these equations. Let b be the sum of the matrix parts and c the sum of the vector parts

$$b = \omega_o^2 \begin{bmatrix} I_{zz} - I_{yy} & I_{xy} & 0 \\ I_{xy} & I_{zz} - I_{xx} & 0 \\ -I_{xz} & -I_{yz} & 0 \end{bmatrix} - pC_D a \begin{bmatrix} 0 & r_y n_x & r_z n_x \\ 0 & r_z n_z - r_x n_x & -r_z n_y \\ 0 & -r_y n_z & r_y n_y - r_x n_x \end{bmatrix} \tag{13.42}$$

and

$$c = \begin{bmatrix} -I_{yz} \\ I_{xz} \\ 0 \end{bmatrix} - pC_D a \begin{bmatrix} 0 \\ n_x r_z \\ -n_x r_y \end{bmatrix} \tag{13.43}$$

The angles are found by solving the following equation.

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = b^{-1} c \tag{13.44}$$

where b must be invertible. Example 13.6 solves the equations for the simple geometry. An alternative approach would be to use a search algorithm. This would retain all the

```
i      = [2.5 0.2 0.1;0.2 2.2 0.1;0.1 0.1 0.5]*1e
         -3;
n      = [0;1;0];
mu     = 3.98600436e5;
sMA    = 7000;
v      = sqrt(mu/sMA);
omega  = sqrt(mu/sMA^3);
omSq   = omega^2;
a      = 1;
cD     = 2.7;
r      = [0.1;0.2;-0.1];
rho    = AtmDens2(sMA-6378.165);
p      = 0.5*rho*cD*v^2;

tAC    = -p*a*[0;n(1)*r(3);-n(1)*r(2)];
aA     = -p*a*[0   n(1)*r(2)              n(1)*r(3)
         ;...
                0   n(3)*r(3)-n(1)*r(1) -n(2)*r(3)
                    ;...
                0 -r(2)*n(3)             r(2)*n(2)-r
                    (1)*n(1)];

aG     = omSq*[i(3,3)-i(2,2) i(1,2) 0;i(1,3)  i
         (3,3)-i(1,1) 0;-i(1,3) -i(2,3) 0];

tGC    = omSq*[-i(2,3); i(1,3);0];


ang    = -(aG + aA)\(tAC + tGC);
```

```
ang =

   -0.0533
    0.0472
    0.4289
```

Example 13.6: Torque-equilibrium attitude solution.

nonlinear terms. However, it would not provide much intuition. The result of the linear equation could be used as a starting point for a search.

13.5.3.3 Momentum management with solar wings

If a spacecraft has two solar wings, which is true for most geosynchronous spacecraft, they can be used to remove roll and yaw momentum. Roll is the axis pointing at the Earth and yaw is along the velocity vector. The geometry is shown, for one wing, in Fig. 13.5.

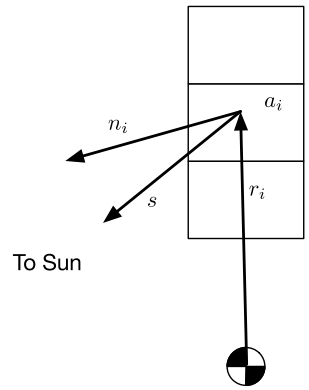


Figure 13.5 Solar-pressure unloading geometry.

If we assume that the solar wings rotate about pitch, γ, and are perfectly flat, then the torque due to the two wings is

$$t = -p \sum_{i=2} N a_i r_i^{\times} \left(s^T n_i (2(\rho_s + \rho_d/3) n_i + (\rho_a + \rho_d) s \right) \tag{13.45}$$

where p is the solar pressure, which is the solar flux divided by the speed of light, a_i is the panel area, assumed to be the same for both wings, s is the Sun vector, n is the unit normal to the surface, A is the area of the surface, S is the solar flux in N/m^2, ρ_s is the specular reflection coefficient, ρ_a is the absorbed radiation coefficient, and ρ_d is the diffuse reflection coefficient. The center-of-mass is assumed to be at the origin.

Example 13.7 shows the effect of rotating the solar wings together or separately. The torques and angles are at the top of each figure.

Some spacecraft add a tab at the base of the wing that is perpendicular to the wing plane. That can easily be accommodated in this analysis by adding additional panels with the appropriate areas, normals, and position vectors.

13.5.3.4 Gravity-gradient momentum management

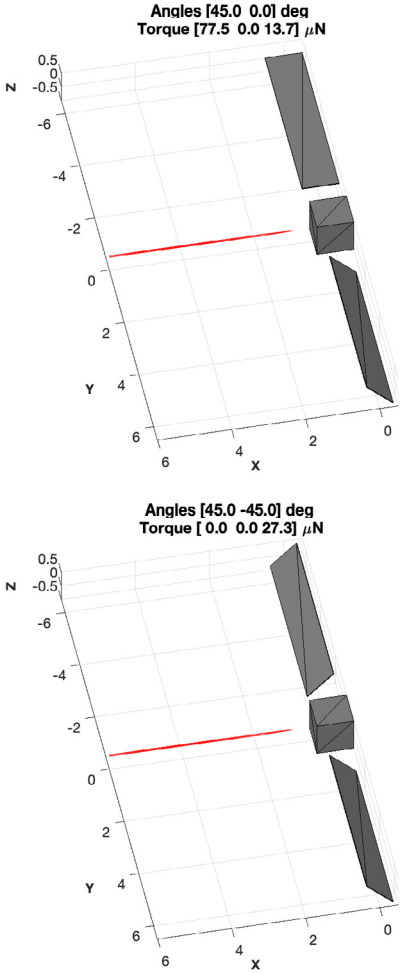
A gravity gradient can also be used to unload momentum. The gravity-gradient torque is

$$T_x = \omega_o^2 \left[(I_{zz} - I_{yy})\alpha + I_{xy}\beta - I_{yz} \right] \tag{13.46}$$

$$T_y = \omega_o^2 \left[(I_{zz} - I_{xx})\alpha + I_{xy}\beta + I_{xz} \right] \tag{13.47}$$

$$T_z = -\omega_o^2 \left[I_{xz}\alpha + I_{yz}\beta \right] \tag{13.48}$$

```
p        = 1367/3e8;
rho      = [0.3; 0.7; 0];
u        = [0;0;1];

ang      = [45   45;...
             0  -45];

l        = 5; % Wing length  (m)
w        = 1; % Wing width  (m)

% Vertices and faces for drawing
[vP,fP] = Box(w,l,0.02);
[vB,fB] = Box(1,1,1);

 % Position vectors
rP       = [0; l/2+1.5;0];
rN       = [0;-l/2-1.5;0];

area     = w*l;

torque   = zeros(3,size(ang,2));

for k = 1:size(ang,2)
  bP       = [CosD(ang(1,k)) 0 -SinD(ang(1,k));...
              0              1  0;...
              SinD(ang(1,k)) 0  CosD(ang(1,k))];

  bN       = [CosD(ang(2,k)) 0 -SinD(ang(2,k));...
              0              1  0;...
              SinD(ang(2,k)) 0  CosD(ang(2,k))];
  forceP         = SolarF(p,rho,u,bP'*u,area);
  forceN         = SolarF(p,rho,u,bN'*u,area);
  torque(:,k)  =  Cross(rP,forceP) +...
                  Cross(rN,forceN);

  % For drawing
  v        = vB;
  f        = fB;

  f        = [f;fP+size(v,1)];
  vW       = vP;
  vW(:,2) = vW(:,2) + 4;
  vR       = (bP*vW')';
  v        = [v;vR];
  f        = [f;fP+size(v,1)];
  vW       = vP;
  vW(:,2) = vW(:,2) - 4;
  vR       = (bN*vW')';
  v        = [v;vR];

  s1 = sprintf('Angles [%4.1f %4.1f] deg\n',ang
       (:,k));
  s2 = sprintf('Torque [%4.1f %4.1f %4.1f] \\muN'
       ,torque(:,k)*1e6);

  DrawObject(v,f,[s1 s2]);
  view([0.2 1 1])
  title(s)

  Arrow3D(0.02,5,[6;0;0],[-1;0;0],[1 0 0],0.2);
end
```



Example 13.7: Solar-pressure momentum control.

In matrix form

$$\omega_0^2 \begin{bmatrix} I_{zz} - I_{yy} & I_{xy} & 0 \\ I_{zz} - I_{xx} & I_{xy} & 0 \\ -I_{xz} & -I_{yz} & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} T_x + \omega_o^2 I_{yz} \\ T_y - \omega_o^2 I_{xz} \\ T_z \end{bmatrix} \tag{13.49}$$

It is not possible to unload T_z. We can unload x and y,

$$\omega_0^2 \begin{bmatrix} I_{zz} - I_{yy} & I_{xy} \\ I_{zz} - I_{xx} & I_{xy} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} T_x + \omega_o^2 I_{yz} \\ T_y - \omega_o^2 I_{xz} \end{bmatrix} \tag{13.50}$$

Analytically, invert the equation and we get

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{\begin{bmatrix} I_{xy} & -I_{xy} \\ I_{xx} - I_{zz} & I_{zz} - I_{yy} \end{bmatrix}}{\omega_o^2 I_{xy}(I_{xx} - I_{zz})} \begin{bmatrix} T_x + \omega_o^2 I_{yz} \\ T_y - \omega_o^2 I_{xz} \end{bmatrix} \tag{13.51}$$

where x and y momentum can be unloaded as long as the denominator is not zero. Example 13.8 gives an example.

```
% 3U inertia
i     = [2.5 0.2 0.1;0.2 2.5 0.1;0.1 0.1 0.5]*1e
    -3
mu    = 3.98600436e5;
sMA   = 7000;
omega = sqrt(mu/sMA^3)
omSq  = omega^2;
t     = [1e-6;1e-7];
ang   = (1/omSq)*[i(3,3)-i(2,2) i(1,2);...
                  i(3,3)-i(1,1) i(1,3)]...
                  \[t + omSq*[i(2,3);-i(1,3)]]
```

```
omega = 0.0011
ang = 1.0e-07 *[0.0047;0.1046]
```

Example 13.8: Gravity-gradient momentum unloading.

CHAPTER 14

Attitude estimation

14.1. Introduction

This chapter describes how to design static attitude estimators using measurements from star cameras, planet sensors, Sun sensors, magnetometers, GPS receivers, and other devices. The algorithms in this chapter produce an estimate based on one set of data, though noise filtering may be employed. Planet sensors, Sun sensors, and magnetometers produce two-axis measurements. A GPS receiver, with suitable antennas, can produce three-axis attitude information. The chapter on Sensors provides additional information.

Chapter 15 shows how to put attitude estimation in a recursive framework. This allows you to rigorously incorporate uncertainty into the attitude estimation process.

14.2. Star sensor

First, get unit vectors for each valid star k in the field of view,

$$u_{M_k} = \text{Unit}\left(\begin{bmatrix} \frac{p_{x_k}}{f} \\ \frac{p_{y_k}}{f} \\ 1 \end{bmatrix}\right) \tag{14.1}$$

where p_x and p_y are the focal-plane coordinates and f is the focal length. Fig. 14.1 shows the pinhole-camera optical diagram and the pixel plane. The star images are defocused to spread them among multiple pixels. This allows for subcentroid location accuracy.

At least three centroids are needed. More centroids will reduce the errors due to noise. Find the corresponding catalog unit vectors, u_C through star identification. Make a 3-by-n matrix of the vectors. For example, with four stars the matrix is

$$u_M = \begin{bmatrix} u_{M_1} & u_{M_2} & u_{M_3} & u_{M_4} \end{bmatrix} \tag{14.2}$$

Then, compute the transformation matrix with the pseudoinverse,

$$m = u_M \text{pinv}\,(u_C) \tag{14.3}$$

You then convert m to an attitude quaternion that will orthonormalize it.

https://doi.org/10.1016/B978-0-32-399915-1.00025-5

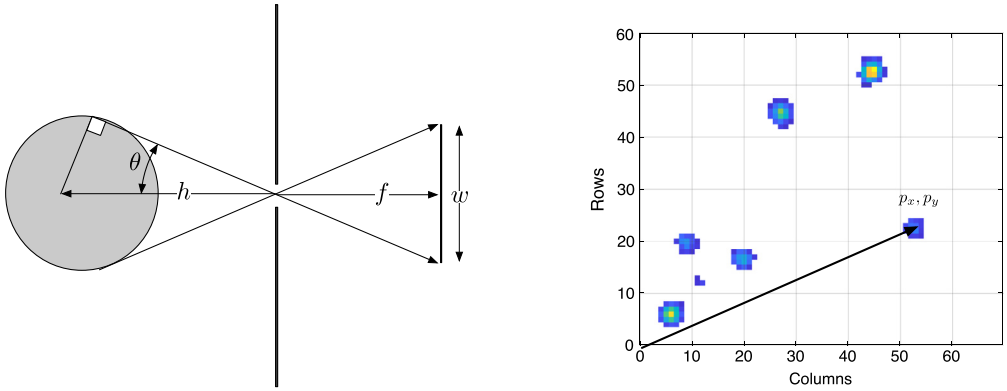


Figure 14.1 Star-camera geometry. The diagram on the left shows the pinhole camera. The image on the right shows a focal plane.

14.3. Planet sensor

14.3.1 Acquisition

Chapter 11 describes the mathematical basis for planet-sensor measurements. Both static and scanning sensors must be pointed at the Earth (or planet) to get a valid measurement. Since most sensors have a limited field of view, it is necessary to have an alternative way to find the planet when the sensor cannot see its target. For a momentum-bias satellite, with its momentum vector normal to the orbit, this is just a matter of pitching the spacecraft (rotating around orbit normal) until the planet is in view. For a three-axis satellite, a search may be needed. A magnetometer can provide a two-axis attitude that can be used to find the planet if there is an onboard magnetic-field model and the orbit position is known. A Sun sensor can also be used for this purpose.

14.3.2 Roll and pitch measurements from a planet sensor

Static planet sensors have a fixed ring of thermal sensors that skirt the horizon of the planet. The output is proportional to the amount of the sensor that sees the planet. If the sensor sees cold space the output is, ideally, zero. If it is entirely on the planet its output is one. Fig. 14.2 shows a sensor with four detectors 90 degrees apart. The field of view is set by the operational altitude of the sensor. The figure shows two cases; one with zero roll and the other with 0.6 degrees of roll.

The chapter on Machine Intelligence shows how a neural network can be trained to convert sensor outputs into roll and pitch. In this chapter, we create a simple algorithm that uses two sensors for roll and two for pitch.

Example 14.1 computes the output from a static Earth sensor. The last plot is the fit to the data. As can be seen, the outputs of the sensors are linearly related to roll and

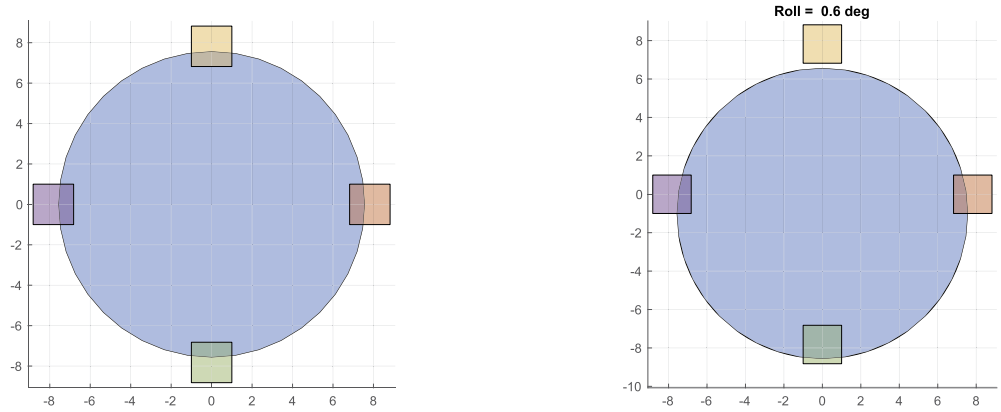


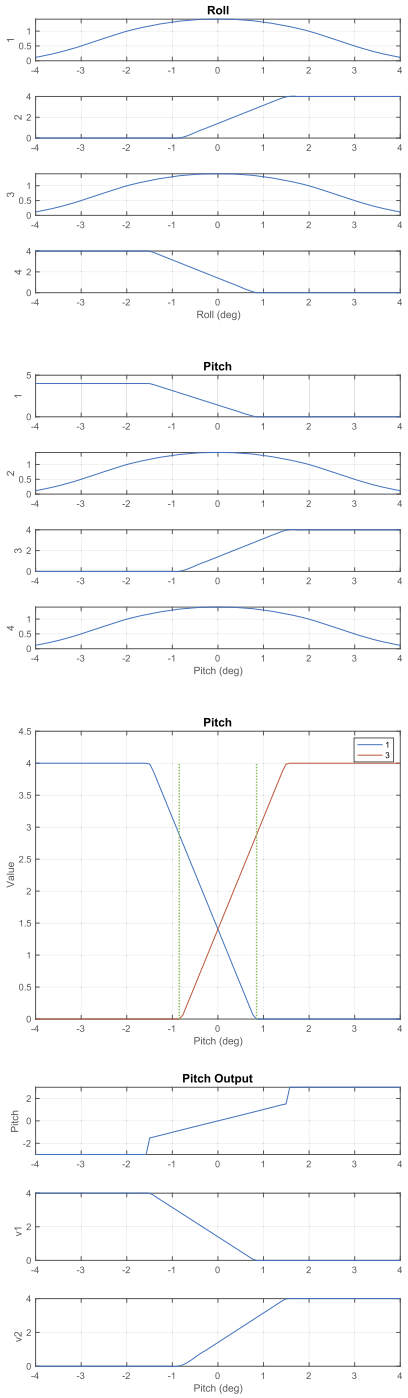
Figure 14.2 Earth sensor at zero roll and 0.6 deg roll.

pitch so a simple algorithm will suffice. The first plot shows the four sensor outputs for a roll sweep. Sensors 2 and 4 are linearly related to roll when they are on the edge of the Earth. Sensors 1 and 3 give outputs over a range of roll but do not give sign inputs. In addition, the outputs are nonlinear. The second plot shows a similar pattern for pitch, only with sensors 1 and 3. The third plot shows sensors 1 and 3 for pitch. The last plot shows a fit to pitch using sensor 1 and 3 data only.

The fit sets the output to 3 with the sign based on whether the pitch is negative or positive. When both v1 and v2 are valid it averages their outputs.

```
% Out of range
pitch(v1>=3.99)=-3;
pitch(v3>=3.99)= 3;

% 4 is -1.52 2.74 is = +0.79
j = find(v1<3.99);
pitch1(j) = -1.52+(4-v1(j))*(1.52+0.79)/4;

% 4 is -1.52 2.74 is = -0.79
j = find(v3<3.99);
pitch3(j) = 1.52+(v3(j)-4)*(1.52+0.79)/4;

j = find(v1<3.99 & v3 ==0);
pitch(j) = pitch1(j);
j = v3<3.99 & v1 ==0;
pitch(j) = pitch3(j);
j = find(v3 > 0 & v1 > 0);
```

```
%% Earth Sensor

d = StaticEarthSensor;

d.az = [0 pi/2 pi 3*pi/2];
d.el = (pi/20)*ones(1,4);


r    = [0;0;42167];
StaticEarthSensor([1;0;0;0],r,d);


n = 100;
y= zeros(4,n);

q = [1;0;0;0];

% Roll
angle = linspace(-4,4);
for k = 1:n
  q       = Sa2Q([angle(k)*pi/180;0;0]);
  y(:,k) = StaticEarthSensor(q,r,d);
end

Plot2D(angle,y,'Roll (deg)', {'1' '2' '3' '4'}, '
       Roll')

roll = 0.01;
StaticEarthSensor(QUnit([1;roll;0;0]),r,d);
s = sprintf('Roll = %4.1f deg',roll*180/pi);
title(s)

% Pitch
for k = 1:n
  q       = Sa2Q([0;angle(k)*pi/180;0;0]);
  y(:,k) = StaticEarthSensor(q,r,d);
end

Plot2D(angle,y,'Pitch (deg)', {'1' '2' '3' '4'},
       'Pitch')
Plot2D(angle,y([1 3],:),'Pitch (deg)', {'Value'},
       'Pitch')

x = [-0.848 -0.848];
line(x,[0 4],'linestyle',':','linewidth',2,'color
       ',[0 1 0])

x = [0.848 0.848];
line(x,[0 4],'linestyle',':','linewidth',2,'color
       ',[0 1 0])
legend('1','3')

p = SensorToPitch(y(1,:),y(3,:));
Plot2D(angle,[p;y([1 3],:)],'Pitch (deg)',{'Pitch
       ' 'v1' 'v2'},'Pitch Output')
```



Example 14.1: Static Earth-sensor outputs and a pitch-angle fit to outputs 1 and 3.

```
pitch(j) = 0.5*(pitch1(j) + pitch3(j));
```

A similar algorithm, using v2 and v3 can be used for roll.

14.4. Sun sensor

Since a single sensor is just a cosine detector it cannot give direction. If two such sensors are mounted together as shown in Fig. 14.3 the direction can be found. Assume that

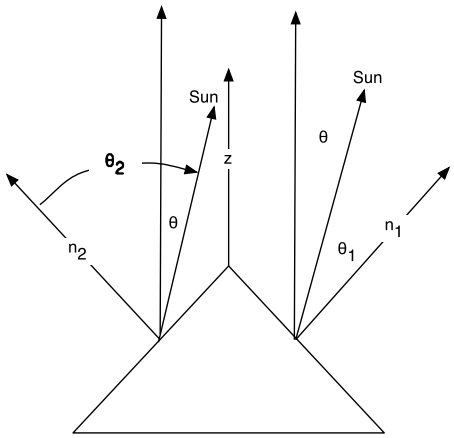


Figure 14.3 Simple one-dimensional Sun sensor.

the output of the sensor is

$$y = \cos\theta \tag{14.4}$$

The angles are

$$\theta_1 = \cos^{-1}(n_1^T u_{Sun}) \tag{14.5}$$
$$\theta_2 = \cos^{-1}(n_2^T u_{Sun}) \tag{14.6}$$

If the angle between z and the sensor normal is β, then

$$\theta_1 = \theta + \beta \tag{14.7}$$
$$\theta_2 = \beta - \theta \tag{14.8}$$

so that

$$\theta = \frac{\theta_2 - \theta_1}{2} \tag{14.9}$$

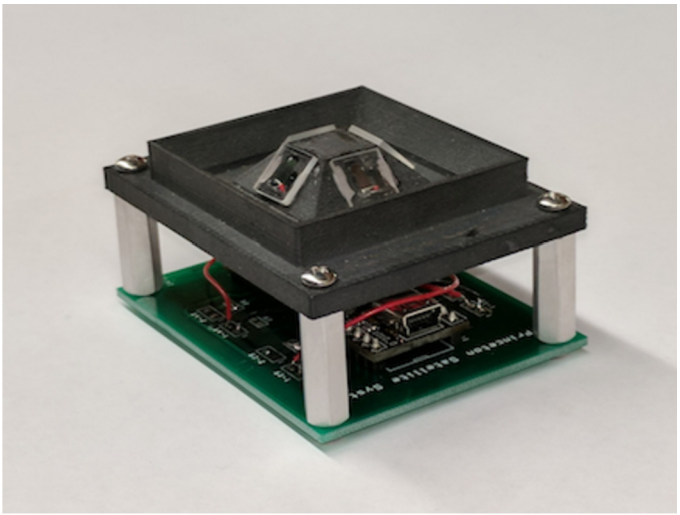


Figure 14.4 Two-axis analog Sun sensor. The supporting structure is 3D printed. The windows are visible on the top.

which gives sign information. Fig. 14.4 shows a pyramidal sensor that can measure two axes. This sensor has an analog-to-digital converter and uses USB as its digital interface.

More generally, the output of a sensor is modeled by a cosine series

$$y = \sum_{k=1}^{N} a_k \cos^k \theta \tag{14.10}$$

where a_k is specified by the vendor. If a_k for $k > 1$ are small, this can be solved using a Newton–Raphson algorithm to find the root of $f(x) = 0$. The algorithm is iterative.

$$\delta = -\frac{f(x)}{f'(x)} \tag{14.11}$$

$$x = x + \delta \tag{14.12}$$

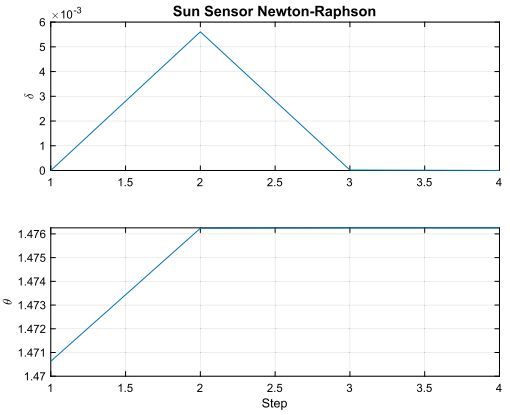
Continue this until δ is small. In this case

$$f(\theta) = y - \sum_{k=1}^{N} a_k \cos^k \theta \tag{14.13}$$

$$f'(\theta) = -\sum_{k=1}^{N} k a_k \sin\theta \cos^{k-1} \theta \tag{14.14}$$

Example 14.2 shows how quickly the Newton–Raphson method converges. As is shown, one iteration is sufficient.

```
a = [1 0.6 0.3];
y = 0.1;
n = 4;

delta    = zeros(1,n);
theta    = zeros(1,n);
theta(1) = acos(y)/a(1);

for k = 2:n
  c        = cos(theta(k-1));
  s        = sin(theta(k-1));
  f        = y - (a(1)*c + a(2)*c^2 + a(3)*c^3);
  fD       = s*(a(1) + 2*a(2)*c + 3*a(3)*c^2);
  delta(k) = -f/fD;
  theta(k) = theta(k-1) + delta(k);
end

Plot2D(1:n,[delta;theta],'Step',{'\delta' '\theta'},'Sun Sensor Newton-Raphson')
set(gca,'xlim',[1 n]);

c = cos(theta(end));
s = sin(theta(end));
err = y - (a(1)*c + a(2)*c^2 + a(3)*c^3)
```



Example 14.2: Newton–Raphson method for Sun-angle computation.

14.5. Magnetometer

A magnetometer gives the magnetic-field vector in the body frame. If an ephemeris is available with a magnetic-field model, then the angle between the measured vector and the ECI magnetic-field vector can be found. In Section 14.7 we show how to combine the magnetic-field vector with another vector to get the ECI to body quaternion. For just the angle you first compute the magnetic field for the current orbital position, transform that into the ECI frame and then take the inverse cosine of the dot product. The process for getting the ephemeris magnetic field vector is shown in Fig. 14.5.

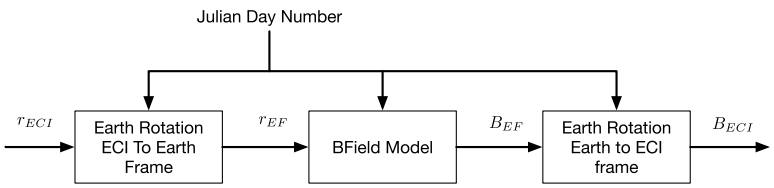


Figure 14.5 ECI magnetic-field computation.

The angle is

$$\theta = \cos^{-1}(u_{ECI}^T u_{body}) \tag{14.15}$$

This defines a cone around the magnetic field vector. Another measurement is needed to find the location on the cone.

14.6. GPS

GPS and other navigation satellites send signals that are used to determine position and velocity. Each signal gives range and range rate. Four signals are needed to resolve the three-dimensional position and velocity. This same system can be used for attitude determination [1,2]. The concept is shown in Fig. 14.6.

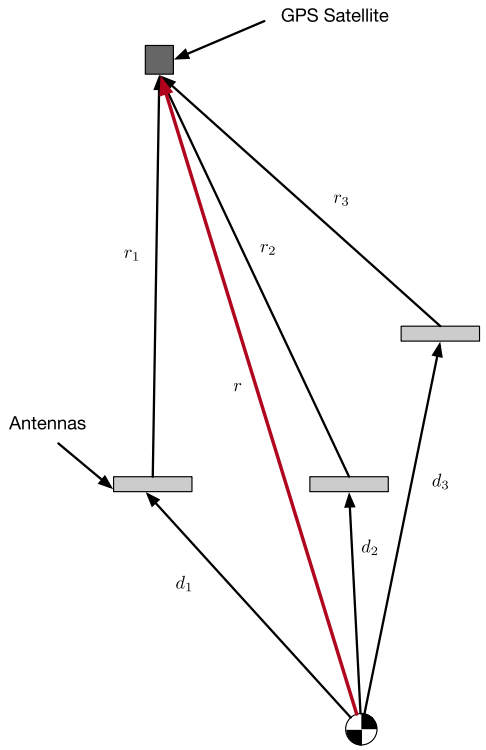


Figure 14.6 GPS attitude geometry. The range signals will be a function of the attitude. Multiple GPS satellites are used.

GPS attitude determination uses the time difference of the double-differenced carrier-phase measurements of the GPS signals. The attitude precision that can be achieved from a GPS attitude determination system is limited by the multipath signals and the baseline length between the antennas [3]. For this analysis, we will assume that the measurements of r and r_k have sufficient precision. The equation for each vector for the jth GPS satellite is

$$r_j + A(r_{jk} + d_k) = 0 \tag{14.16}$$

where A transforms from the inertial to the body frame. A is the attitude matrix we wish to compute. r is known from the GPS orbit determination. d_k is also known. The

measurements are just out of range, so

$$r_{jk} = \rho_{jk} A u_{jk} \tag{14.17}$$

where ρ_k is the measured range and u_k is the unit vector pointing from the antenna to the GPS satellite measured in the body frame. The vector equation is now

$$A(r_{jk} + d_k) = r_j \tag{14.18}$$

Assume that all of the vectors are aligned with r and the unit vector for r is u. Then, the equation becomes

$$A(\rho_{jk} u_j + d_k) = r_j \tag{14.19}$$

If we have at least three measurements, we can compute A. Let $b_{kj} = \rho_{kj} u_j + d_k$.

$$A = F\text{pinv}(B) \tag{14.20}$$

where the columns of B are b_{kj} and for each column kj of B there is a corresponding column of F that is r_j. A is most easily orthonormalized by converting it to a quaternion. Example 14.3 gives an example. You need to have multiple satellites for a good solution, as shown in the script. ans is the true quaternion. This is not an issue since GPS requires at least four satellites for accurate orbit determination.

```
q    = QRand;
a    = Q2Mat(q);

rGPS = 22000*Unit(randn(3,5));

rSat = [0;0;7000];
d    = [1 0 -1;1 -1 1;0 0 0];

r    = rGPS - rSat;
i    = 0;

b    = zeros(3,3);
for j = 1:size(r,2)
  rS = a'*r(:,j);
  u  = Unit(rS);
  for k = 1:3
   i      = i + 1;
   rK     = rS - d(:,k);
   rho    = Mag(rK);
   b(:,i) = rho*u + d(:,k);
   f(:,i) = r(:,j);
 end
end

aC = f*pinv(b);

% Orthnormalze
Mat2Q(aC)
q
```

```
ans =
1.921206024595309e-01   -
2.403235884576184e-01
5.256707101347798e-01   -
7.930965479134801e-01
q =
1.921221216347452e-01   -
2.403228320128406e-01
5.256708682857887e-01   -
7.930978281571304e-01
```

Example 14.3: GPS attitude solution.

14.7. Earth/Sun/magnetic field

It is possible to measure the ECI to body quaternion using two vector measurements from a combination of Earth sensors, Sun sensors, and magnetometers. Fig. 14.7 shows the geometry.

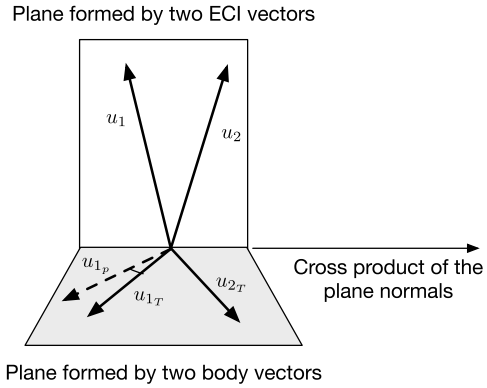


Figure 14.7 Vector geometry for Earth/Sun/magnetic-field-attitude estimation.

First, unitize the measured and cataloged vectors. Then, compute the normals for each plane.

$$n = u_1 \times u_2 \tag{14.21}$$

$$n_T = u_{1_T} \times u_{2_T} \tag{14.22}$$

Unitize both n and n_T. Create a quaternion that rotates n into n_B,

$$a = n \times n_T \tag{14.23}$$

$$d = n^T n_T \tag{14.24}$$

$$s = \sqrt{2(1+d)} \tag{14.25}$$

$$q_T = \begin{bmatrix} \frac{1}{2}s \\ \frac{c}{s} \end{bmatrix} \tag{14.26}$$

$$u_{1_P} = q \otimes u_1 \tag{14.27}$$

Create a quaternion that rotates u_{1_P} into u_{1_T},

$$a = u_{1_P} \times u_{1_T} \tag{14.28}$$

$$d = u_{1_P}^T u_{1_T} \tag{14.29}$$

$$s = \sqrt{2(1+d)} \tag{14.30}$$

$$q_P = \begin{bmatrix} \frac{1}{2}s \\ \frac{c}{s} \end{bmatrix} \tag{14.31}$$

The ECI to body quaternion is

$$q = q_T \otimes q_P \tag{14.32}$$

Any combination of nadir vectors, magnetic-field vectors, and Sun vectors can be used. You need to know the ECI references for these vectors. The body-unit vectors can be obtained from a magnetometer and any two-axis Earth or Sun sensor. The nadir vector can be obtained from GPS or an onboard ephemeris.

14.8. Noise filters

Sensors can be noisy and may need filtering. The order of a filter determines how effective it is at removing noise. The higher the order, the faster the roll-off. This comes with the price of increased lag in the loop. If the cutoff frequency is near the control bandwidth, it will be necessary to adjust the control-algorithm parameters.

Let us work in the continuous domain. We use a proportional-derivative controller with a first-order filter. The PD controller is

$$\frac{u}{\theta} = K(1 + \tau_r s) \tag{14.33}$$

where K is the forward gain and τ_r is the derivative time constant. The first-order filter is

$$\frac{\theta}{\theta_m} = \frac{1}{\tau_f s + 1} \tag{14.34}$$

At low frequencies this has a gain of 1. At high frequencies the filter is

$$\frac{\theta}{\theta_m} = \frac{1}{\tau_f s} \tag{14.35}$$

The response decreases linearly as s increases. This can also be written in state-space form

$$\dot{x} = -\frac{1}{\tau_f}x + \frac{1}{\tau_f}u \tag{14.36}$$

$$y. = x \tag{14.37}$$

The total transfer function is the series of the first-order filter and the PD controller.

$$\frac{u}{\theta_m} = K\left(\frac{1 + \tau_r s}{1 + \tau_f s}\right) \tag{14.38}$$

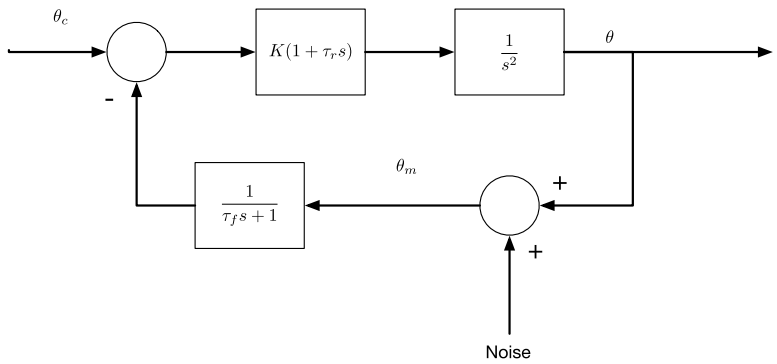
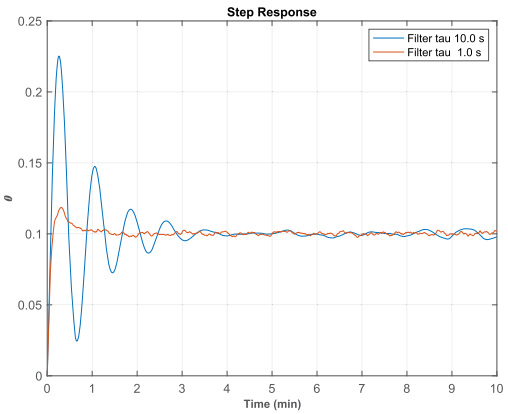


Figure 14.8 The closed-loop system block diagram.

```
%% First order filter

omegaN    = 0.1;
zeta      = 2;
tauR      = 2*zeta/omegaN;
kF        = omegaN^2;
thetaC    = 0.1;
oneSigma  = 0.01;
tauF      = tauR*[0.5  0.05];
n         = 6000;
dT        = 0.1;
xP        = zeros(2,n);
s         = cell(2);

for j = 1:2
  d.tauF = tauF(j);
  s{j}   = sprintf('Filter tau %4.1f s',d.tauF);
  x      = [0;0];
  errOld = 0;
  thetaF = 0;
  for k = 1:n
    thetaM    = x(1) + oneSigma*randn;
    thetaF    = thetaF + dT*(thetaM - thetaF)/
          tauF(j);
    err       = thetaF - thetaC;
    u         = -kF*(err + tauR*(err-errOld)/dT);
    errOld    = err;
    xP(j,k)   = x(1);
    x         = RK4(@RHS,x,dT,0,u);
  end
end

t = (0:n-1)*dT;

TimeHistory(t,xP,{'\theta'},'Step Response'
      ,{1:2});
legend(s{1},s{2})

%% Right hand side
function xDot = RHS(x,~,u)
omega     = x(2);
xDot      = [omega;u];
end
```



Example 14.4: First-order filter response for two different filter time constants.

Assume we want a damping ratio of 1 and a bandwidth of 0.1 rad/s. Assume that the plant is a double integrator, $\frac{1}{s^2}$, then $K = \omega_n^2$ and $\tau_r = \frac{2}{\omega_n}$. Let us look at the controller response with a filter cutoff at twice the control-system bandwidth and five times the control-system bandwidth. The system block diagram is shown Fig. 14.8.

Example 14.4 shows the response with two different first-order filter time constants. The controller and filter are implemented with a first-order hold. The filter with the smaller time constant, and higher filter cutoff, passes more noise but has better damping. The filter with the larger time constant has very low damping but the signal is cleaner.

An nth-order Butterworth filter is often a reasonable choice for noise filtering. Butterworth filters are called "maximally flat" filters because they do not have any ripple in the stop band or the pass band. This makes them well suited for control systems. Peaking is not always undesirable, because it reduces the phase lag. An even order (2nd, 4th etc.) can be assembled by putting 2nd-order state-space blocks in series. One block is

$$\dot{x} = ax + bu \tag{14.39}$$

$$y = cx + du \tag{14.40}$$

$$a = \begin{bmatrix} 0 & \omega_C \\ -\omega_C & -2\zeta\omega_C \end{bmatrix} \tag{14.41}$$

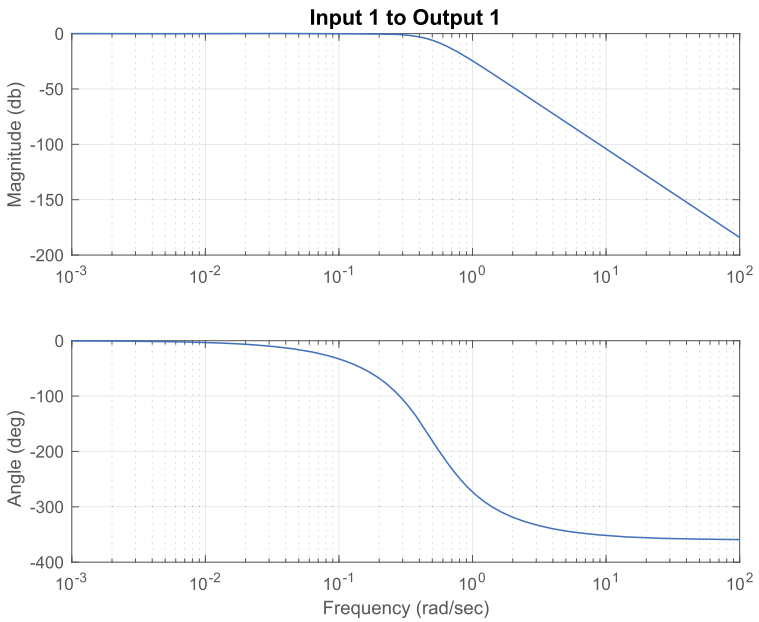


Figure 14.9 Butterworth fourth-order filter with $\omega_C = 0.5$.

$$b = \begin{bmatrix} 0 \\ \omega_C \end{bmatrix} \tag{14.42}$$

$$c = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{14.43}$$

$$d = 0 \tag{14.44}$$

where $\zeta = \frac{\sqrt{2}}{2}$ and ω_C is the cutoff frequency. An example of a fourth-order Butterworth filter, found by calling CButter(4, 0.5, 1) is shown in Fig. 14.9.

References

[1] Charles Wang, Rodney A. Walker, Werner Enderle, Single antenna attitude determination for FedSat, in: Proceedings Institute of Navigation GPS-02, 2002, pp. 1–12.
[2] G. Lu, Development of a GPS Multi-Antenna System for Attitude Determination, Ph.D. thesis, University of Calgary, 1995.
[3] D. Kim, R. Langley, GPS RTK-Based Attitude Determination for the e-POP Platform onboard the Canadian CASSIOPE Spacecraft in Low Earth Orbit, 01 2007.

CHAPTER 15

Recursive attitude estimation

15.1. Introduction

This chapter describes how to design recursive attitude estimators. Attitude estimators are algorithms that can be used to determine the orientation of a spacecraft from sensor data.

There are two types of recursive estimators, batch and sequential. Batch estimators are used when you have collected a large span of data and want to process it at once. Sequential estimators are used when you want to use the data as it is collected.

15.2. Batch methods

Batch methods of attitude determination take a set of measurements over multiple time samples and attempt to fit a solution. There are four types of batch least-squares estimators.

1. least squares;
2. weighted least squares;
3. maximum likelihood;
4. Bayesian.

Given the measurement equation

$$z = Hx + v \tag{15.1}$$

where v is unbiased white noise, the least-squares estimator is

$$\hat{x} = (H^T H)^{-1} H^T z \tag{15.2}$$

where z may be a vector with measurements from different types of sensors taken at different times. The assumption is that the state being measured has not changed much during the interval. This process treats all measurements equally.

The next type of estimator is the weighted least squares with a weighting matrix R

$$\hat{x} = (H^T R^{-1} H)^{-1} H^T R^{-1} z \tag{15.3}$$

R may be chosen by any means desired. For example, if there are too many measurements from one sensor, rather than removing them completely one might deweight them.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00026-7

The third kind of estimator is the maximum likelihood estimator. It is

$$\hat{x} = (H^T R^{-1} H)^{-1} H^T R^{-1} z \tag{15.4}$$

which is identical in form to the weighted least squares. The difference is that R is the covariance matrix for z. Thus, the selection of the weighting matrix is put on a firm statistical footing.

The final type of estimator is the Bayesian estimator which is

$$\hat{x} = (P_0^{-1} H^T R^{-1} H)^{-1} H^T R^{-1} z \tag{15.5}$$

where P_0 is the a priori covariance matrix for x. This assumes that we know something about x prior to taking any measurements.

It is instructive to look at the case of three sequential measurements. In this case, the measurement equation is

$$z = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{15.6}$$

The least-squares estimator becomes

$$\hat{x} = \frac{1}{3}(z_1 + z_2 + z_3) \tag{15.7}$$

which is just the average of the three measurements. The weighted least-squares (and maximum likelihood) estimators become

$$\hat{x} = \frac{\left(\frac{z_1}{r_1} + \frac{z_2}{r_2} + \frac{z_3}{r_3}\right)}{\frac{1}{r_1} + \frac{1}{r_2} + \frac{1}{r_3}} \tag{15.8}$$

If the measurements happen to be identical, the weights have no affect on the estimate. Otherwise, the measurement with the smallest weight will be favored. The Bayesian estimator is

$$\hat{x} = \frac{\left(\frac{z_1}{r_1} + \frac{z_2}{r_2} + \frac{z_3}{r_3}\right)}{\frac{1}{r_1} + \frac{1}{r_2} + \frac{1}{r_3} + \frac{1}{p_0}} \tag{15.9}$$

In the limit, as p_0 goes to infinity, the estimate goes to zero.

The Differential Corrector algorithm is

$$x_{k+1} = x_k + \left[S_0 + h_k^T W H_k\right]^{-1} \left[H_k^T W(\gamma - h_k) + S_0(x_0 - x_k)\right] \tag{15.10}$$

where W is the measurement weighting matrix and S_0 is the a priori covariance matrix of vector x_0. For the purposes of evaluating the quality of the result the condition number of the inverse should be evaluated. S_0 tends to improve the conditioning.

The conjugate-gradient method finds the solution to the loss function directly. The loss function is

$$2J = (y - h_k)^T W (y - h_k) + (x_0 - x_k)^T S_0 (x_0 - x_k) \tag{15.11}$$

Conjugate gradient and steepest descent both require the derivative of J, which requires H. Both find x_k that minimizes J.

15.3. Vector measurements

For small angles the transformation matrix from the reference frame to the body frame is given by

$$M = E - \theta^{\times} \tag{15.12}$$

where E is a 3-by-3 identity matrix and $\theta^{\times}$ is the skew-symmetric matrix

$$\theta^{\times} = \begin{bmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{bmatrix} \tag{15.13}$$

The angle definitions are shown in Fig. 15.1.

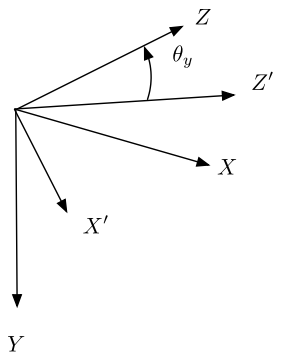


Figure 15.1 Small angles.

Assume that the transformation matrix from the body frame to the sensor frame is m_S. Then, a vector in the reference frame becomes

$$u_M = M_S (E - \theta^{\times}) u \tag{15.14}$$

$$u_M = M_S u - M_S \theta^{\times} u \tag{15.15}$$

$$u_M - M_S u = M_S \gamma^{\times} \theta \tag{15.16}$$

$$z = h\theta \tag{15.17}$$

where E is the identify matrix and M_S transforms to the sensor frame. This is the form of the standard linear measurement equation for a Kalman-filter formulation. h is a 3-by-3 matrix, however, the inverse of h is singular; hence it is not possible to estimate three components of θ from any set of measurements, z, which are derived from a single vector unless h (and therefore u) is time varying. If multiple, independent vectors measurements are available, the three components of θ can be estimated.

Vector measurements include star measurements, Sun measurements, planet measurements, and magnetic-field measurements. All can be combined using this formulation.

15.4. Disturbance estimation

It is often advantageous to estimate disturbance torques on a spacecraft. A simple example is a momentum bias geosynchronous satellite that is Earth-pointing. The equations of motion for roll/yaw become

$$-\frac{T_z}{h_w} = \dot{\theta}_x - \omega_o \theta_z \tag{15.18}$$

$$\frac{T_x}{h_w} = \dot{\theta}_z + \omega_o \theta_x \tag{15.19}$$

where h_w is the momentum wheel momentum, T_x is the roll torque, T_z is the yaw torque, and ω_o is orbit rate. A proportional control could be designed using these equations. The state-space matrices are

$$A = \begin{bmatrix} 0 & \omega_o \\ -\omega_o & 0 \end{bmatrix} \tag{15.20}$$

$$B = \begin{bmatrix} 0 & -\dfrac{1}{h_w} \\ \dfrac{1}{h_w} & 0 \end{bmatrix} \tag{15.21}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{15.22}$$

$$D = \begin{bmatrix} 0 & 0 \end{bmatrix} \tag{15.23}$$

$$u = \begin{bmatrix} T_x \\ T_z \end{bmatrix} \tag{15.24}$$

In this case, we only have a roll-angle measurement. Note that the torques crosscouple so that the yaw torque drives roll and the roll torque drives yaw. The disturbances on the geosynchronous spacecraft are due to solar pressure. The spacecraft has solar arrays that always point at the Sun and the bus points at the Earth. Thus there will be an inertially fixed torque on the spacecraft (which in the body frame is at orbit rate), a body-fixed torque, and harmonics of orbit rate. If we limit our estimator to the body-fixed torque, orbit rate, and twice orbit rate, the augmented A matrix is

$$A = \begin{bmatrix} 0 & \omega_o & -\frac{1}{h_w} & -\frac{1}{h_w} & -\frac{1}{h_w} & 0 & 0 \\ -\omega_o & 0 & 0 & 0 & 0 & \frac{1}{h_w} & \frac{1}{h_w} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\omega_o & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2\omega_o \\ 0 & 0 & 0 & \omega_o & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2\omega_o & 0 & 0 \end{bmatrix} \tag{15.25}$$

The first row is yaw, with roll-torque and yaw-angle inputs. The second row is roll, with roll-angle and yaw-torque inputs. The third row is the roll-bias torque. The fourth row is the roll orbit-rate disturbance. The fifth row is the roll twice per orbit disturbance torque. The sixth row is the yaw once per orbit disturbance while the seventh row is the yaw twice per orbit disturbance. The yaw-bias torque is not included since it cannot be observed from a roll measurement. The estimator is comprised of the low-order roll/yaw dynamics model augmented with a disturbance model. The disturbance model assumes that the cyclic roll and yaw disturbances have the same magnitude and are 90 deg out of phase. That is why the same differential equation for the orbit rate and twice orbit rate disturbance dynamics can produce both estimated yaw- and roll-torque outputs.

Since the idea is to produce estimates of the disturbances, the estimator is modified to include known torque inputs from the control system

$$\dot{\hat{x}} = A\hat{x} + L(\phi - H\hat{x}) + Gu_c \tag{15.26}$$

$$y = C\hat{x} \tag{15.27}$$

Using full-state feedback, with no additional control dynamics,

$$u_C = -K\hat{x} \tag{15.28}$$

where K is the gain matrix and the new estimator equation becomes

$$\dot{\hat{x}} = (A - GK)\hat{x} + L(\phi - H\hat{x}) \tag{15.29}$$

$$y = C\hat{x} \tag{15.30}$$

which means that the new estimator plant is that for the closed-loop system. The spacecraft is now expected to behave as if it had the ideal control system operating. Normally, this would be done by replacing the plant matrix with the closed-loop plant matrix.

15.5. Stellar-attitude determination

15.5.1 Introduction

Attitude determination systems measure the angular offset between a spacecraft reference frame and the desired reference frame. The simplest attitude determination systems measure the misalignment directly. For example, an Earth sensor measures the roll and pitch errors directly. Yaw can be measured directly using a magnetometer. These systems measure the attitude with respect to an Earth-oriented frame. Errors with respect to inertial frames can be measured using the Sun or stars as references. Two types of recursive attitude determination systems are:

1. Recursive with gyros;
2. Recursive without gyros.

15.5.2 Gyro-based attitude determination

Gyros are often part of an attitude determination system. Gyros measure inertial angular rates in the body frame. Rate gyros output the rate and rate-integrating gyros output the integrated rate. The latter measurements are known as angular increments. To use a gyro as an attitude reference requires that at some point in time the gyro be initialized. For example with a rate-integrating gyro, the output is set to zero, and the current orientation of the spacecraft is measured by other means. This is the attitude reference. The integrated rates are then used to determine the attitude with respect to the initialized value. Since the outputs are integrated body rates, the readout cannot be used directly unless the spacecraft does not rotate very much.

For spacecraft-attitude determination, the model used in the Kalman filter is the error model for the gyros, not the dynamics and kinematics of the spacecraft. In essence, the gyros are assumed to be a perfect attitude reference except for biases (which may drift). The gyro outputs are not considered measurements. This greatly simplifies the model in the estimator. It is rarely feasible to accurately model a spacecraft in sufficient fidelity to use such a model in the estimator. Sometimes an approximation, such as a low-pass filter, represents the spacecraft when a gyro is not available.

The Kalman filter propagates two quantities. One is the state, i.e., the attitude and gyro-bias estimates. The second is the state covariance, a measure of the uncertainty of the estimates. The covariance estimate will often reach a steady state in which case the Kalman gain (that is computed the covariance) remains fixed. Sometimes, the covariance propagation is omitted and the steady-state gains used in a fixed-gain Kalman filter.

Kalman filters that are used for stellar-attitude determination are usually extended Kalman filters or unscented Kalman filters. This means that the state and measurement matrices are functions of the state. The measurement matrix changes as different stars are used and since the state propagation is nonlinear, the state-transition matrix also changes as a function of the state.

The best representation of the attitude is the quaternion that is a four-parameter set such that

$$1 = q_0^2 + q_1^2 + q_2^2 + q_3^2 \tag{15.31}$$

Propagation of the quaternion is singularity free. The q_0 component is sometimes called the scalar component and the other three are the vector component. The cost of the set being singularity free is that more than the three minimum states are needed to represent the attitude. If we were to use the following as our state vector

$$x = \begin{bmatrix} q \\ b \end{bmatrix} \tag{15.32}$$

our corresponding covariance matrix would by 7×7, but the quaternion covariances would not be independent. The standard approach is to use the error quaternion.

It is easiest to start with transformation matrices. The small-angle rotation matrix from the unrotated frame to the rotated frame is

$$M_\delta = \begin{bmatrix} 1 & \theta_z & -\theta_y \\ -\theta_z & 1 & \theta_x \\ \theta_y & -\theta_x & 1 \end{bmatrix} \tag{15.33}$$

This matrix can be written as

$$M_\delta = E - \theta^\times \tag{15.34}$$

where E is the identity matrix and $\theta^\times$ is the skew-symmetric matrix

$$\theta^\times = \begin{bmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{bmatrix} \tag{15.35}$$

The transformation matrix from the inertial frame to the body frame (which we want to estimate) is

$$M_{ib} = \hat{M}_{ib} M_\delta \tag{15.36}$$

The equivalent quaternion expression is

$$q_{ib} = \hat{q}_{ib} \otimes q_\delta \tag{15.37}$$

The small-rotation matrix can be converted to a quaternion using the following equations

$$q_\delta = \frac{1}{2}\begin{bmatrix} 2 \\ \frac{M_\delta(3,2)-M_\delta(2,3)}{2} \\ \frac{M_\delta(1,3)-M_\delta(3,1)}{2} \\ \frac{M_\delta(2,1)-M_\delta(1,2)}{2} \end{bmatrix} \tag{15.38}$$

or

$$q_\delta = \begin{bmatrix} 1 \\ -\frac{\theta_x}{2} \\ -\frac{\theta_y}{2} \\ -\frac{\theta_z}{2} \end{bmatrix} \tag{15.39}$$

or

$$q_\delta = \begin{bmatrix} 1 \\ \delta \end{bmatrix} \tag{15.40}$$

where

$$\delta = -\frac{1}{2}\begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} \tag{15.41}$$

We then can propagate

$$x = \begin{bmatrix} \delta \\ b \end{bmatrix} \tag{15.42}$$

and maintain a 6×6 covariance matrix.

The entire algorithm is as follows. Start with the standard Kalman filter,

$$k = ph^T(hph^T + r)^{-1} \tag{15.43}$$
$$x = x + k(z - hx) \tag{15.44}$$
$$p = khph^T k^T + krk^T \tag{15.45}$$
$$x = \Phi x \tag{15.46}$$
$$p = \Phi p \Phi^T \tag{15.47}$$

The first three equations are the measurement incorporation and the last two are the state update. h will depend on the number of stars so the size of h and r will vary as different numbers of stars are used as part of the measurement. The number of stars does not change the size of the covariance matrix.

For each star the measurements are the location of the star centroid in the focal plane. Assume that we have a star with unit vector u in the inertial frame. Assume that

the sensor boresight is along the z-axis. Then, the measurement for the star is

$$u_m = M_{bs} M_{ib} u_c \tag{15.48}$$

where m_{bs} transforms from the body frame to the sensor frame, m_{ib} transforms from the inertial frame to the body frame, u_c is the catalog star unit vector, u_m is the measured unit vector, and m_{ib} is the matrix equivalent of q. Our measurement is the first two elements of u_m. We can expand m_{ib} as the product of the nominal attitude and the error. We can then write this equation as

$$u_m = M_{bs} \hat{M}_{ib} M \delta u_c \tag{15.49}$$

This expands to

$$u_m = M_{bs} \hat{M}_{ib} u_c - M_{bs} \hat{M}_{ib} \theta^{\times} u_c \tag{15.50}$$

so that

$$h = M_{bs} M_{ib} u_c^{\times} \tag{15.51}$$

and the actual measurement matrix is the upper 2×3 part of h.

The state-transition matrix, Φ that updates x must also be computed. This is used only to update p and is not used to propagate the attitude state or the biases.

The equation for the quaternion time derivative is

$$\dot{q} = \frac{1}{2} \begin{bmatrix} 0 & -\omega^T \\ \omega & -\omega^{\times} \end{bmatrix} q \tag{15.52}$$

If we write the equation for the true quaternion (dropping the *ib* subscript)

$$\dot{q} = \frac{1}{2} \Omega(\omega) q \tag{15.53}$$

and the equation for the estimate is

$$\dot{\hat{q}} = \frac{1}{2} \Omega(\hat{\omega}) \hat{q} \tag{15.54}$$

If we define a four-element rate vector as

$$\omega = \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{15.55}$$

then the rate equations can be written as

$$\dot{q} = -\frac{1}{2}\omega \otimes q \tag{15.56}$$

and

$$\dot{\hat{q}} = -\frac{1}{2}\hat{\omega} \otimes \hat{q} \tag{15.57}$$

The delta quaternion is found from

$$\delta = q \otimes \hat{q}^T \tag{15.58}$$

and

$$q = \delta \otimes \hat{q} \tag{15.59}$$

The derivative is

$$\dot{\delta} = \dot{q} \otimes \hat{q}^{-1} + q \otimes \dot{\hat{q}}^{-1} \tag{15.60}$$

Substituting for the rates we find

$$\dot{\delta} = \frac{1}{2}\left[\omega \otimes q \otimes \hat{q}^{-1} + q \otimes (\hat{\omega} \otimes \hat{q})^{-1}\right] \tag{15.61}$$

Substituting for q we get

$$\dot{\delta} = \frac{1}{2}\left[\omega \otimes \delta \otimes \hat{q} \otimes \hat{q}^{-1} + \delta \otimes \hat{q} \otimes (\hat{\omega} \otimes \hat{q})^{-1}\right] \tag{15.62}$$

Applying the identity

$$(\hat{\omega} \otimes \hat{q})^{-1} = -\hat{q}^{-1} \otimes \hat{\omega} \tag{15.63}$$

we get

$$\dot{\delta} = \frac{1}{2}\left[\omega \otimes \delta - \delta \otimes \hat{\omega}\right] \tag{15.64}$$

The rates are related by the equation

$$\hat{\omega} = \omega + b - \hat{b} \tag{15.65}$$

Therefore

$$\dot{\delta} = \frac{1}{2}\left[\hat{\omega} \otimes \delta - \delta \otimes \hat{\omega}\right] + \frac{1}{2}(\hat{b} - b) \otimes \delta \tag{15.66}$$

If we neglect nonlinear terms, and redefine

$$\delta = \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} \tag{15.67}$$

we can write

$$\dot{\delta} = \frac{1}{2}\hat{\omega} \times \delta + \frac{1}{2}\beta \tag{15.68}$$

where β is the bias error, just as δ is the quaternion error.

Fig. 15.2 shows how q flows through the system.

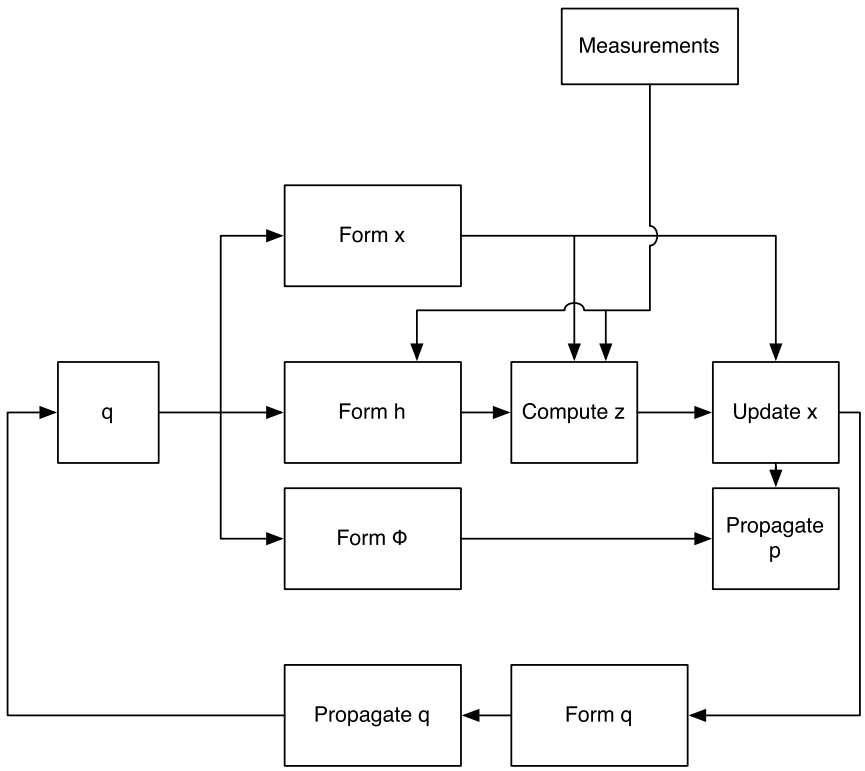


Figure 15.2 Flow of the quaternion through the Kalman filter.

15.5.3 A single-axis Kalman filter with a gyro

A simple sensing geometry is shown in the following Fig. 15.3. The sensors are used to

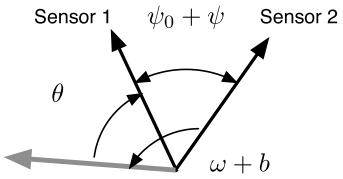


Figure 15.3 Two-sensor, single-axis sensing.

correct for the drift of the gyro that measures angular rate directly. The two sensors are separated by a known angle and an unknown bias.

The equations are

$$\dot{\theta} = \omega + b + \eta_\theta \tag{15.69}$$

$$b = \eta_b \tag{15.70}$$

$$\psi = \eta_\psi \tag{15.71}$$

$$\gamma_1 = \theta + \eta_\gamma \tag{15.72}$$

$$\gamma_2 = \theta + \psi + \eta_\gamma \tag{15.73}$$

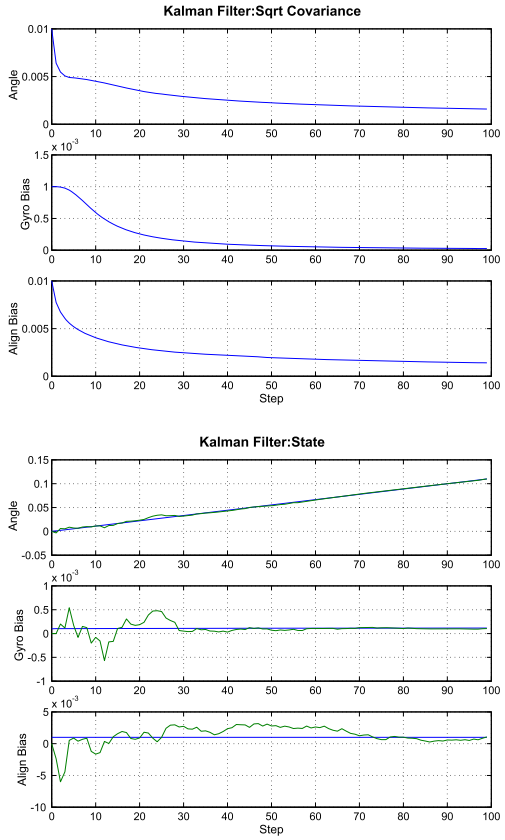
The gyro-angle derivative is the sum of the true rate, the bias, and noise. The derivative of the gyro bias equals white noise. The derivative of the bias angle between the two angle sensors is also white noise. The two measurements are the sum of the true angle and noise. However, the second measurement includes the unknown bias.

This set of equations is linear time invariant and can be implemented using a standard Kalman filter. The results are shown in Example 15.1. The attitude estimate tracks the true attitude very well. The bias estimate converges quickly but the estimate of the angle bias between the two sensors is slow to converge.

These results are reflected in the covariance square roots. The covariance plots show the square roots of the diagonal covariance elements and are representative of the expected error in the estimates. The angle covariance decreases and begins to converge. The covariance tells us that the attitude estimate will track the true attitude, despite the presence of the gyro bias. The gyro-bias estimate reflects this as it goes to zero. However, the alignment bias does not drop as quickly, meaning that we expect that there will be a bias in the estimate.

The accuracy of the estimates will depend on the relative measurement and state noise. In this demo, the noise and measurement covariances are derived directly from the noise numbers used in the simulation so, given the numbers in the simulation, these are the best results that can be obtained.

```
a  = [0 1 0;0 0 0;0 0 0];
b  = [1;0;0];
c  = [1 0 0;1 0 1];
dT = 1;

[a, b] = C2DZOH( a, b, dT );

nSim = 100;

xPlot = zeros(3,nSim);
ePlot = zeros(3,nSim);
pPlot = zeros(3,nSim);

omega = 0.001;

gyroBias      = 1e-6;
alignmentBias = 1e-7;
measurement   = 1e-2;
angle         = 1e-5;

x    = [0;omega/10;0.001];
xEst = [0;0;0];
r    = measurement^2*eye(2);
q    = diag([angle^2,gyroBias^2,alignmentBias^2]);
p    = diag([.01;.001;.01].^2);
z    = zeros(3,1);

for k = 1:nSim
  y                  = c*x + measurement*randn(2,1);
  xPlot(:,k)         = x;
  ePlot(:,k)         = xEst;
  pPlot(:,k)         = sqrt(diag(p));
  [xEst, p, g, z, pY] = KFilter( r, a, q, c, xEst, y, p, [], b*omega );
  x                  = a*x + b*omega + [angle; gyroBias; alignmentBias].*randn(3,1);
end

n = 0:(nSim-1);
yLbl = strvcat('Angle','Gyro Bias','Align Bias');
Plot2D(n,[xPlot;ePlot],'Step',yLbl,'Kalman Filter:State','lin',['[1 4]';'[2 5]';'[3 6]'])
Plot2D(n,pPlot,'Step',yLbl,'Kalman Filter:Sqrt Covariance')
PrintFig(1,1,1,'KFStateAndEstimates');
PrintFig(1,1,2,'KFSRC');
```



Example 15.1: Kalman filter.

15.5.4 Star identification

It is necessary to determine which stars are visible in the field-of-view. A complete attitude determination system is illustrated Fig. 15.4.

Star identification is a major problem with any system that uses stars for attitude or orbit determination. Star cameras are used on manned spacecraft and ships for attitude determination. In these applications, they rely on the ability of a crew member to identify the stars that he or she sees and then to inform the computer of the stars' identities. Very accurate and reliable attitude determination is possible. This kind of man-in-the-loop system relies on the ability of people to rapidly identify patterns.

A practical star-identification system must deal with the issues in Table 15.1.

More details about star identification are given in Appendix K.

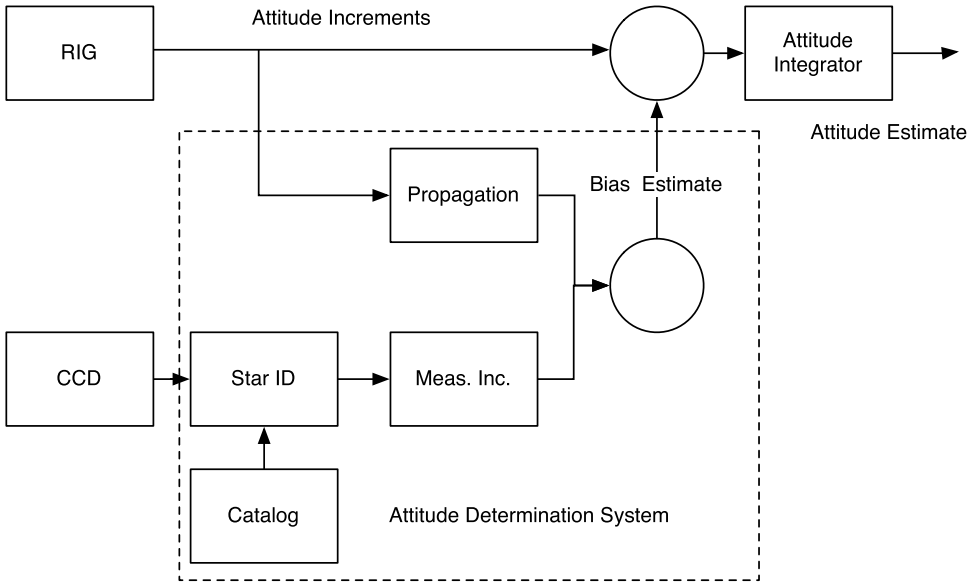


Figure 15.4 Stellar attitude determination system.

Table 15.1 Star identification.

Issue	Description
Sensor calibration	The sensing elements of the star camera may not have the same response to starlight. Thus if a star image is measured by pixel k, the resulting intensity may not be the same as when it is measured by pixel j. While the sensor can be calibrated, and a calibration map stored as part of the algorithm, the calibration may change with time.
Sensor noise	Pixels may give finite output (even after calibration) when there is no star image on the pixel.
Sensor resolution	The number of pixels are limited. Resolution usually is improved by smearing the image over several pixels. Resolution improvements of up to 100 are possible. The problem with finite resolution is that a measured dot product may agree with many catalog dot products. Thus it is necessary to look at patterns.
Other objects	The Sun, planets, moons, other spacecraft, and debris may appear in the field of view. Other objects may completely obscure the star image or appear as false stars. The latter is a more serious problem. The former is solved either by running gyros all the time or having a second-star tracker.
Stray light	Even the slightest Sun reflection that enters the camera will overwhelm the star images. Baffles are used to minimize this problem.
Speed	As an example, the Hipparcos catalog has over 100 000 stars. If the entire catalog is used, in a lost-in-space condition, comparisons need to be made with all of those stars. That can take some time.

15.6. Kalman filter with roll, pitch, and yaw and a gyro

For this Kalman filter we need roll, pitch, and yaw. These can be obtained by an Earth sensor and magnetometer. The first step to computing these angles is to compute the local vertical/local horizontal (LVLH) quaternion

$$q_{LVLH} = f(r, v) \tag{15.74}$$

where r and v are the ECI position and velocity vector. From the magnetic-field model get the unit vector of the magnetic field in the LVLH frame.

$$u_B = q_{LVLH} \otimes B_{ECI} \tag{15.75}$$

We need the unit nadir vector in the LVLH frame

$$u_N == q_{LVLH} \otimes B_{ECI} \frac{r}{|r|} \tag{15.76}$$

These two vectors are then used to produce a quaternion from body to LVLH

$$q_\delta = f(u_B, u_N) \tag{15.77}$$

q_δ is then converted into Euler angles. This is the measurement.

The state matrix is

$$\Phi = \Delta T \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{15.78}$$

The input matrix is

$$\Gamma = \Delta T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{15.79}$$

The measurement matrix is

$$h = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{15.80}$$

The measurement incorporation step is

$$k = \frac{pb}{hph^T + r} \tag{15.81}$$

$$x = x + k(\gamma - hx) \tag{15.82}$$

$$p = (E(6) - kh)p \tag{15.83}$$

where p is the state covariance, r is the measurement covariance, $E(6)$ is a six-by-six identity matrix, and γ is the measurement.

The state-propagation input is the gyro angle

$$\omega = \frac{\theta - \theta_{old}}{\Delta T} \tag{15.84}$$

$$x = \Phi x + \Gamma\omega \tag{15.85}$$

$$p = \Phi p \Phi^T + Q \tag{15.86}$$

$$\theta_{old} = \theta \tag{15.87}$$

where Q is the plant covariance matrix. It contains all uncertainties in the gyro model.

15.7. Kalman filter with a quaternion measurement

The Kalman filter estimates the gyro bias and the ECI to body quaternion. It has two parts, state prediction and measurement incorporation. The gyro model in state-space form is

$$x_k = ax_{k-1} + b\omega_{k-1} \tag{15.88}$$

$$\gamma_k = hx_k \tag{15.89}$$

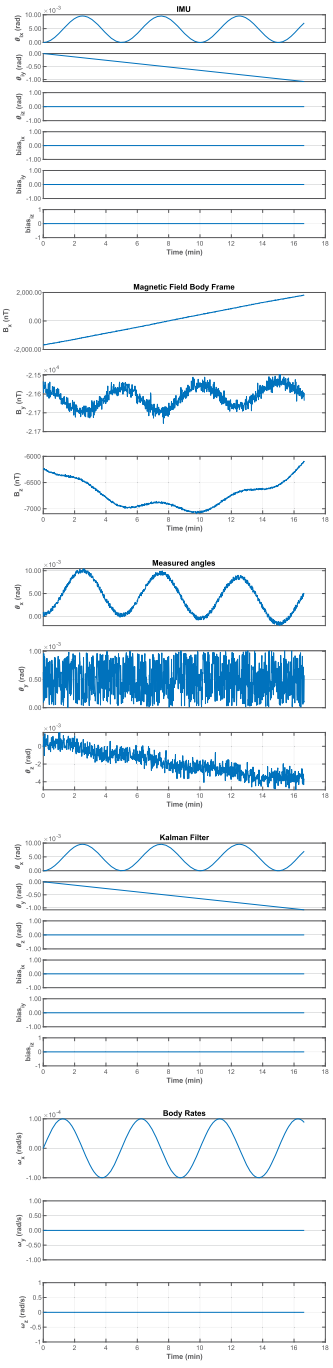
where

$$x_k = \begin{bmatrix} \theta_x \\ \theta_y \\ \theta_z \\ b_x \\ b_y \\ b_z \end{bmatrix} \tag{15.90}$$

The matrices are

$$a = \begin{bmatrix} 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & \Delta t \end{bmatrix} \tag{15.91}$$

```
dKF          = GyroKalmanFilter;
dRHS         = RHSGyroWithOrbit;
jD0          = Date2JD([2041 7 5]);
bias         = [0;0;0];
iMUAngle     = [0;0;0];
el           = [7000 0 0 0 0 0];
q            = [1;0;0;0];
n            = 1000;
dT           = 1;
bNoise       = 1e-8;
measNoise    = 1e-3;
dOmega       = [0.0001;0;0];
omega0       = 2*pi/300;

% Initalize the Kalman Filter
dKF.r        = measNoise^2*eye(3);
dKF.q        = [diag(dRHS.ang1Sig.^2) zeros(3,3)
        ;...
                zeros(3,3)                diag(dRHS.
                    b1Sig.^2)];
dKF.p        = 0.1*dKF.q;

%% Simulate
[r,v]        = El2RV(el);
orbRate      = 2*pi/Period(el(1));
q            = QLVLH(r,v);
x            = [r;v;q;iMUAngle;bias];
secInDay     = 86400;
dKF.omega    = [0;-orbRate;0];

% Allocate memory
xP   = zeros(length(x)+15,n);

% Create a time vector
t    = (0:n-1)*dT;
jD   = jD0 + t/secInDay;

% Simulation loop
for k = 1:n

  % Body rate
  dRHS.omega = [0;-orbRate;0] + dOmega*sin(omega0
        *t(k));

  % Convenient to use local variables
  r = x(1: 3);
  v = x(4: 6);
  qECIToBody = x(7:10);
  iMUAngle = x(11:13);

  % Sensor measurements
  bMeas = QForm(qECIToBody,BDipole(r,jD(k))) +
        bNoise*randn(3,1);
  nadirMeas = Unit(-QForm(qECIToBody,r));

  % Angles with respect to LVLH
  meas = AngleDetermination(r,v,nadirMeas,bMeas,
        jD(k)) + measNoise*rand(3,1);

  % Kalman Filter
  dKF = GyroKalmanFilter(meas,iMUAngle,dKF);

  % Do one simulation step
  xP(:,k) = [x;bMeas*1e9;meas;dKF.x;dRHS.omega];
  x = RK4(@RHSGyroWithOrbit,x,dT,0,dRHS);
end

% Put plot labels in a cell array
yL = {'x (km)'  'x (km)' 'x (km)',...
      'v_x (km/s)' 'v_y (km/s)' 'v_z (km/s)', ...
      'q_s' 'q_x' 'q_y' 'q_z' ...
      '\theta_{ix} (rad)' '\theta_{iy} (rad)' '\
            theta_{iz} (rad)'...
      'bias_{ix}' 'bias_{iy}' 'bias_{iz}'...
      'B_x (nT)'  'B_y (nT)'  'B_z (nT)' ...
      '\theta_x (rad)', '\theta_y (rad)', '\
            theta_z (rad)' ...
      '\theta_x (rad)', '\theta_y (rad)', '\
            theta_z (rad)' ...
      'bias_{ix}' 'bias_{iy}' 'bias_{iz}'...
      '\omega_x (rad/s)', '\omega_y (rad/s)', '\
            omega_z (rad/s)'};
% Generate the plots.
tL = {'Position' 'Velocity' 'Quaternion' 'IMU'
      ...
      'Magnetic Field Body Frame' 'Measured
            angles'...
      'Kalman Filter' 'Body Rates' };
kL = {1:3 4:6 7:10 11:16 17:19 20:22 23:28
      29:31};
for j = 1:length(tL)
  k = kL{j};
  TimeHistory(t,xP(k,:),yL(k),tL{j});
end
Figui
```



Example 15.2: Kalman filter with angle measurements and a gyro.

$$b = \begin{bmatrix} \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{15.92}$$

$$h = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{15.93}$$

The Kalman-filter measurement incorporation step is

$$k = \frac{ph^T}{hph^T + r} \tag{15.94}$$

$$x_e = k(y - hx_e) \tag{15.95}$$

$$p = (E - kh)p \tag{15.96}$$

where E is a six-by-six identity matrix, p is the covariance matrix, r is the measurement covariance matrix, and h is the measurement matrix. The measurement is y, which is the delta quaternion

$$\delta q = \left(q_m^T \otimes q_e\right)^T \tag{15.97}$$

where q_m is the measured quaternion and q_e is the current estimate of the quaternion. The input to the measurement update are elements 2 through 4 of δq.

$$u_\omega = \frac{\omega}{|\omega|} \tag{15.98}$$

$$\theta = \frac{|\omega|\Delta t}{2} \tag{15.99}$$

$$\delta q = \begin{bmatrix} \cos\theta & u_\omega \sin\theta \\ -u_\omega \sin\theta & E\cos\theta - (u_\omega \sin\theta)^\times \end{bmatrix} q \tag{15.100}$$

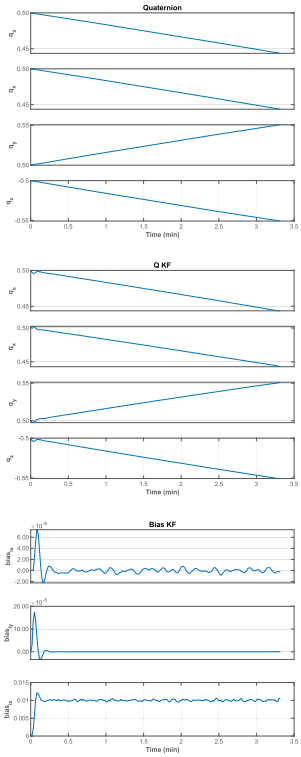
$$p = apa^T + q \tag{15.101}$$

where E is a 3-by-3 identify matrix, a is the state-transition matrix, q is the model covariance matrix, and Δt is the time step. The output of the rate-integrating gyro is differenced to get rate, ω. This is where the gyro connects to the model. IMU angles are treated as a state-prediction input, not a measurement.

The Kalman filter produces an updated delta state. The quaternion part is incorporated as follows. Let δ be elements 2 through 4 of δq

$$n = |\delta|^2 \tag{15.102}$$

```
dKF         = GyroKalmanFilter;
dRHS        = RHSGyroWithOrbit;
jD0         = Date2JD([2041 7 5]);
bias        = [0;0;0.01]; iMUAngle = [0;0;0];
el          = [7000 0 0 0 0 0];
n           = 200;
dT          = 1;
bNoise      = 1e-8;
measNoise   = 1e-3;
dOmega      = [0.0;0.0;0];
omega0      = 2*pi/300;

% Initalize the Kalman Filter
dKF.r = measNoise^2*eye(3);
dKF.p = 0.1*dKF.q;

%% Simulate
[r,v]             = El2RV(el);
orbRate           = 2*pi/Period(el(1));
q                 = QLVLH(r,v);
x                 = [r;v;q;iMUAngle;bias];
secInDay          = 86400;

% Allocate memory for plotting to speed things up
xP  = zeros(length(x)+26,n);

% Create a time vector
t   = (0:n-1)*dT;
jD  = jD0 + t/secInDay;

for k = 1:n
  % Body rate
  dRHS.omega = [0;-orbRate;0] + dOmega*sin(omega0
      *t(k));

  r             = x(1:3);
  v             = x(4:6);
  qECIToBody    = x(7:10);
  iMUAngle      = x(11:13);

  % Sensor measurements
  bECI    = BDipole(r,jD(k));
  bMeas   = QForm(qECIToBody,bECI)...
          + bNoise*randn(3,1);
  nadirMeas = Unit(-QForm(qECIToBody,r));

  % This computes angles with respect to LVLH
  meas = AngleDetermination(r,v,nadirMeas,bMeas,
         jD(k)) ...
       + measNoise*rand(3,1);

  qMeas = TwoUToQ(Unit(-r),bECI,nadirMeas,bMeas);

  % Kalman Filter
  dKF.rECI = r;
  dKF.vECI = v;
  if( k == 1)
    dKF.qECIToBody = qMeas;
  end
  dKF = GyroKalmanFilter(qMeas,iMUAngle,dKF);

  % Do one simulation step
  xP(:,k)   = [x;bMeas*1e9;meas;dKF.x;dRHS.omega;
              qMeas;dKF.qECIToBody;dKF.b];
  x         = RK4(@RHSGyroWithOrbit,x,dT,0,dRHS);
end

% Put plot labels in a cell array
yL = {'x (km)'  'x (km)' 'x (km)',...
'v_x (km/s)' 'v_y (km/s)' 'v_z (km/s)', ...
'q_s' 'q_x' 'q_y' 'q_z' ...
'\theta_{ix} (rad)' '\theta_{iy} (rad)' '\theta_{
      iz} (rad)'...
'bias_{ix}' 'bias_{iy}' 'bias_{iz}'...
'B_x (nT)'  'B_y (nT)'  'B_z (nT)' ...
'\theta_x (rad)', '\theta_y (rad)', '\theta_z (
      rad)' ...
'\theta_x (rad)', '\theta_y (rad)', '\theta_z (
      rad)' ...
'bias_{ix}' 'bias_{iy}' 'bias_{iz}'...
'\omega_x (rad/s)', '\omega_y (rad/s)', '\omega_z
      (rad/s)',...
'q_s' 'q_x' 'q_y' 'q_z' 'q_s' 'q_x' 'q_y' 'q_z'
      ...
'bias_{ix}' 'bias_{iy}' 'bias_{iz}'};

% Generate the plots. Notice how we index a cell
      array.
tL = {'Position' 'Velocity' 'Quaternion' 'IMU'
      ...
      'Magnetic Field Body Frame' 'Measured
          angles'...
      'Kalman Filter' 'Body Rates' 'Q Measured' '
          Q KF' 'Bias KF'};

kL = {1:3 4:6 7:10 11:16 17:19 20:22 23:28 29:31
     32:35 36:39 40:42};
for j = 1:length(tL)
  k = kL{j};
  TimeHistory(t,xP(k,:),yL(k),tL{j});
end
```



Example 15.3: Kalman-filter performance. The quaternion is tracked accurately and the bias is estimated.

$$\delta q_1 = \frac{16 - n}{16 + n} \tag{15.103}$$

$$\rho = \frac{1}{4}(1 + \delta q_1)\delta \tag{15.104}$$

$$\delta_q = \begin{bmatrix} \delta q_1 \\ \rho \end{bmatrix} \tag{15.105}$$

$$q = q \otimes \delta_q \tag{15.106}$$

δq must be normalized prior to the quaternion multiplication.

Example 15.3 shows the Kalman-filter response. It tracks the true quaternion very well and estimates the gyro bias accurately.

CHAPTER 16

Simulation

16.1. Space story

Quote from a simulation engineer, "The only thing you can be certain about with an analog computer is that the answer it produces will be different tomorrow."

16.2. Introduction

Simulations are a critical part of spacecraft development. In many industries, experimentation is the key to developing new technologies. For example, circuit designers build prototype circuits on breadboards, which are easy to change and do not require soldering. Changing component values is easy. A wide range of hardware, such as voltmeters, oscilloscopes, etc. is available for testing. Once the designer is happy with the circuit, a printed-circuit board is built and a soldered version is produced. This process continues until production.

In the earliest days of aviation, flight test was the major method for proving new designs. However, even back in the days of the Wright brothers, the idea of a wind tunnel caught on where engineers could study aerodynamics without killing themselves. Testing new aircraft still had to be done by flying the aircraft, which was a very dangerous occupation.

Many industries used scale models to test out ideas. Automobile manufacturers build wood versions of cars before prototyping. CAD software has not replaced scale models.

Spacecraft presented a whole new problem. First, most spacecraft were one-offs, that is, often you only built one. Exceptions were the crewed spacecraft of the 1960s and 1970s and the ISS transfer vehicles. Even when you were able to fly a series of satellites, like DMSP, the amount of data you could get back from the flight was limited. In any case, you always wanted the first one to work as advertised due to the cost of getting it into space. Testing satellites on the ground before a flight is difficult. Some companies use air-bearing tables, with up to three degrees-of-freedom, but they still do not approach actual spaceflight. Building sensor sources, like star fields, planetary horizons, and the Sun, become whole projects in and of themselves.

Before the advent of the digital computer, the first thing one could do is find analytical solutions. For control systems, a huge number of tools were developed for analyzing control systems. Bode, root-locus, Nyquist plots, etc. provide insight into the performance of a linear system. Nonlinear systems could be analyzed using describing functions. Graphical methods were particularly useful because they would provide

https://doi.org/10.1016/B978-0-32-399915-1.00027-9

a range of information, rather than just a number. For any system, analytical solutions were valued. Mission planning used, and still uses, analytical methods like Kepler propagation and Lambert's law. Patch conics is a way of doing multibody orbit trajectories by hand. Drafting was a critical element that allowed designers to visualize their designs.

Even analytical methods required a lot of computing. Slide rules sped things up but inevitably engineering organizations had to hire rooms full of people, known as computers to do all the number crunching. One of the important math lessons in elementary and high school was learning to use tables of trigonometric functions that engineers, and computers, had to employ.

Analog computers, using electrical circuits as analogs to physical systems, were one approach. Suppose we have a first-order differential equation we wish to simulate

$$\tau \frac{dx}{dt} + x = u \tag{16.1}$$

where u is applied at $t = 0$. This has an analytical solution

$$x = u\left(1 - e^{\frac{-t}{\tau}}\right) \tag{16.2}$$

Fig. 16.1 shows an operational amplifier configured as an analog integrator.

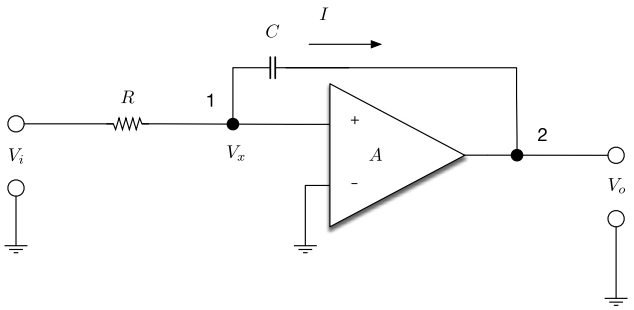


Figure 16.1 Operational amplifier configured as an integrator.

An operational amplifier is a differential amplifier with a very high gain A that amplifies the difference between the plus and minus terminals. The amplifier is so high gain that no current goes into it. The voltage drop across the resistor is

$$V_x - V_i = IR \tag{16.3}$$

The current through the capacitor, C is

$$I = C\frac{d(V_x - V_o)}{dt} \tag{16.4}$$

Since no current is flowing into the op-amp, V_x must be at ground, i.e., 0, where V is the drop across the capacitor. From the voltage drops

$$I = -\frac{V_i}{R} \tag{16.5}$$

or

$$RC\frac{dV_o}{dt} = V_i \tag{16.6}$$

where RC is the time constant, τ. Let $V_o = x$ and $V_i = u$ and we have the integrator part of our differential equation,

$$x_i = \frac{1}{\tau}\int us \tag{16.7}$$

To create our analog computer integrate Eq. (16.1)

$$\tau x + \int x = \int u \tag{16.8}$$

or

$$x = \frac{1}{\tau}\int u - \frac{1}{\tau}\int x \tag{16.9}$$

The analog computer is shown in Fig. 16.2.

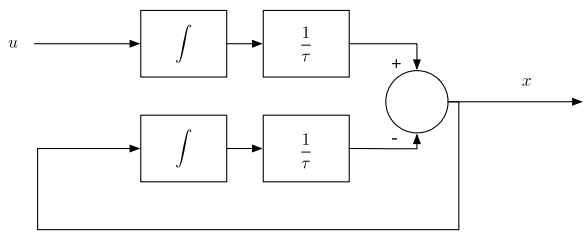


Figure 16.2 Analog computer.

Once you have blocks like integrators, adders, and multipliers you can build a model of any linear system. You can add nonlinear blocks to model nonlinear elements.

The introduction of digital computers made life easier, and harder. The first problem encountered in the 1960s was that the word length was limited, limiting accuracy. The second problem was that floating-point arithmetic was not standard. Not only that, but round-off methods varied and were sometimes done in very strange fashions. Eventually, this was standardized in IEEE-794 [1] by the Institute for Electrical and Electronics Engineers (IEEE), and modern processors when they are made correctly, produce very reliable results.

16.3. Digital simulation

16.3.1 Numerical errors

The analog computer was supplanted by the digital computer in the late 1960s and early 1970s. The fundamental building block of digital simulation is the numerical integrator. Starting with our single integrator

$$\tau \frac{dx}{dt} + x = u \tag{16.10}$$

we can make a numerical simulation as follows.

$$\tau \frac{\Delta x}{\Delta t} + x = u \tag{16.11}$$

$$\Delta x = \frac{\Delta t}{\tau}(u_k - x_k) \tag{16.12}$$

$$x_{k+1} = x_k + \Delta x \tag{16.13}$$

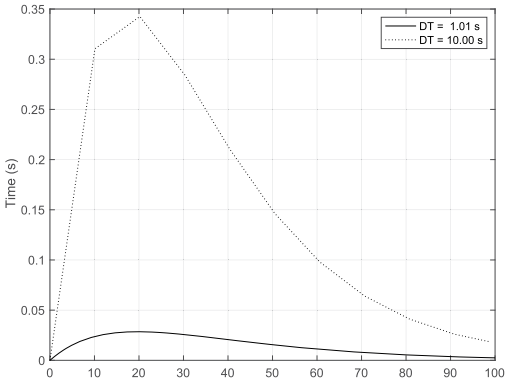
$$x_{k+1} = x_k + \frac{\Delta t}{\tau}(u_k - x_k) \tag{16.14}$$

This simple algorithm is known as Euler's method. This is a discrete version of the continuous differential equation. Δt is the time step and k is the step. Errors arise in two ways. The first is due to the time step. The smaller the time step the more accurate the integration. Therefore there is an error that grows with the size of the time step. Another error is round-off error. Every time an operation is done there will be round-off error since no arithmetic operation is perfectly accurate. The round-off error will grow as more operations are performed. To simulate over a period of time, T requires $n = \frac{T}{\Delta t}$ time steps. The smaller the time step, the more operations and the greater the accumulated round-off error. Smaller time steps reduce truncation error but increase round-off error. With double-precision arithmetic and IEEE standard rounding, round-off error is much less of a problem than it once was. However, one must take care when using an older processor, like the MIL-STD-1750A, which does not use IEEE standard floating-point arithmetic. Appendix A has more on numerical integration methods.

Example 16.1 shows the integration errors when integrating with a one-second time step and a 10-second time step with a time constant, τ of 20 seconds. The simulation is a step response.

Not surprisingly, the 10-second time step has much larger errors. In both cases, the errors are largest when x is changing fast. The simulation is having trouble keeping up with the true state. As time progresses the error drops because the next state is nearly the same as the current state. This shows that one needs to be careful about interpreting simulation results because the numerical errors will be worse when interesting things are happening.

```
1
2 n    = 100;
3 t    = linspace(0,100,n);
4 tau  = 20;
5 u    = 3;
6 x    = u*(1-exp(-t/tau));
7 dT   = t(2);
8 x1   = zeros(1,n);
9 for k = 2:n
10   x1(k) = x1(k-1) + (dT/tau)*(u - x1(k-1));
11 end
12 s1 = sprintf('DT = %5.2f s',dT);
13
14 n = 10;
15 dT = 10;
16 x2 = zeros(1,n+1);
17 for k = 2:n+1
18   x2(k) = x2(k-1) + (dT/tau)*(u - x2(k-1));
19 end
20 s2 = sprintf('DT = %5.2f s',dT);
21
22 NewFig('Simulation')
23 plot(t,abs(x-x1))
24 hold on
25 k = [1:10:100  99];
26 plot(t(k),abs(x(k)-x2))
27 legend(s1,s2)
28 grid
29 ylabel('Error')
30 ylabel('Time (s)')
```



Example 16.1: Euler integration with two different time steps.

16.3.2 Model truncation

We will use a one-axis, two-disk spacecraft to study model truncation error. This is shown in Fig. 16.3. Model truncation may be deliberate, reducing the order of a model to make simulations run faster, or due to modeling errors.

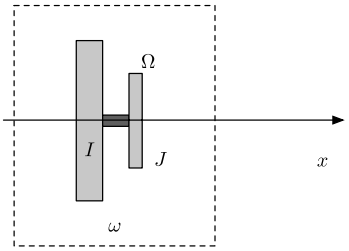


Figure 16.3 One-axis simulation. Two disks are aligned on the same shaft. The dotted line surrounds the entire system.

The angular momentum is

$$H = I\omega + J(\Omega + \omega) \tag{16.15}$$

where I and J are the inertia of the two disks, ω is the angular rate of the entire assembly, and Ω is the angular rate of the second disk with respect to the first. The equations of motion are

$$T = I\dot{\omega} + J(\dot{\omega} + \dot{\Omega}) \tag{16.16}$$

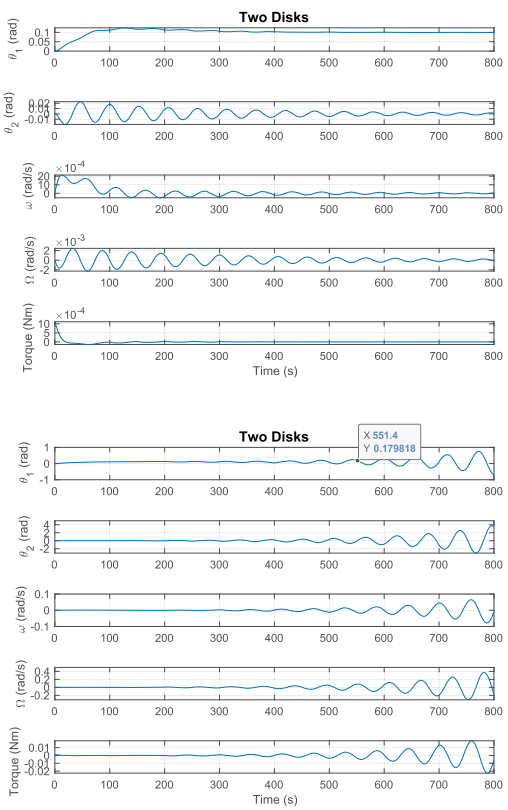
$$T_a = J(\dot{\omega} + \dot{\Omega}) \tag{16.17}$$

where T is the external torque and T_a is the torque between the two disks. T_a is an internal torque. In matrix form this becomes

$$\begin{bmatrix} I+J & J \\ J & J \end{bmatrix} \begin{bmatrix} \dot{\omega} \\ \dot{\Omega} \end{bmatrix} = \begin{bmatrix} T \\ T_a \end{bmatrix} \tag{16.18}$$

Example 16.2 shows the two-disk simulation with a lightly damped connection between the disks. A proportional-integral-derivative (PID) controller is used to do an angle change.

```
n          = 4000;
xP         = zeros(5,n);
dT         = 0.2;
sensorOnBody2 = false;

torque     = 0;
inr1       = 3;
inr2       = 4;
xPID       = [0;0];

x          = [0;0;0;0];
xSet       = 0.1;

[a, b, c, d] = PIDMIMO( inr1+inr2, 0.7071, 0.01,
       1000, 0.1, dT );

dRHS          = struct('inr1',3,'inr2',1,'k',0.01,
       'c',0.001);

for k = 1:n
  if( sensorOnBody2 )
    err = x(1) + x(2) - xSet;
  else
    err = x(1) - xSet;
  end
  xPID       =  a*xPID + b*err;
  dRHS.t1    = -c*xPID - d*err;
  xP(:,k)    = [x;dRHS.t1];
  x          = RK4(@RHS,x,dT,0,dRHS);
end

t  = (0:n-1)*dT;
yL = {'\theta_1 (rad)', '\theta_2 (rad)', '\omega
       (rad/s)'...
       '\Omega (rad/s)' 'Torque (Nm)' };
Plot2D(t,xP,'Time (s)',yL,'Two Disks');

%% Right hand side [ang1;ang2;omega1;omega2]
function xDot = RHS(x,~,d)

inr        = [d.inr1+d.inr2 d.inr2;d.inr2 d.inr2];
t2         = -d.k*x(2) - d.c*x(4);
omegaDot   = inr\[d.t1;t2];

xDot       = [x(3:4);omegaDot];

end
```



Example 16.2: Two disks with a PID controller.

Two cases are shown. In one, the sensor is on the main body and in the second, the sensor is on body 2.

In the first case, the second disk oscillates with respect to the first. The first disk is perturbed by the oscillation. The PID control design assumes that the spacecraft is rigid. The second disk adds some oscillations to the main-body response but does not change the maneuver much. A bigger issue is that a device mounted to the second disk would oscillate. This motion, known as jitter, could degrade the performance of the device.

In the second case, the control system is unstable. Its design does not take into account the flexible connection between actuation and sensing.

In summary, we observe

1. A small degradation in the spacecraft maneuver, if the actuator and sensor are collocated;
2. Instability if the sensor and actuator are not collocated;
3. An oscillation that may impact the payload;
4. Possible structural fatigue.

Changing the model parameters will change the results.

16.4. Applications of simulation

16.4.1 A sequence of simulations for ACS development

Fig. 16.4 shows one possible simulation sequence for the development of an ACS. Maneuver analysis and disturbance analysis are done first to provide inputs to the simulations. Attitude determination, momentum management, and control simulations are done independently and in parallel using simple simulations. The dynamic models need to reflect what is important to each system. For example, if the spacecraft is flexible and the attitude control system must compensate for the flexibility, then the dynamics model for the attitude control simulation must incorporate a flexible-body model. On some spacecraft slosh is significant and then it also needs to be considered. Within the attitude control block, there might be two or three separation simulations. Momentum management generally is just dealing with inertial momentum so a single integrator usually works.

In the next step, momentum management and attitude control are integrated into a single simulation. The highest-fidelity dynamics model from the attitude control block becomes the baseline dynamics model. If the spacecraft has a propulsion system, it is added at this point. Propulsion-system nonlinearities, like minimum impulse bit, etc. are included at this point. The orbit model is added. A point-mass planet model is usually sufficient. This allows engineers to determine the impact of orbital position on attitude control.

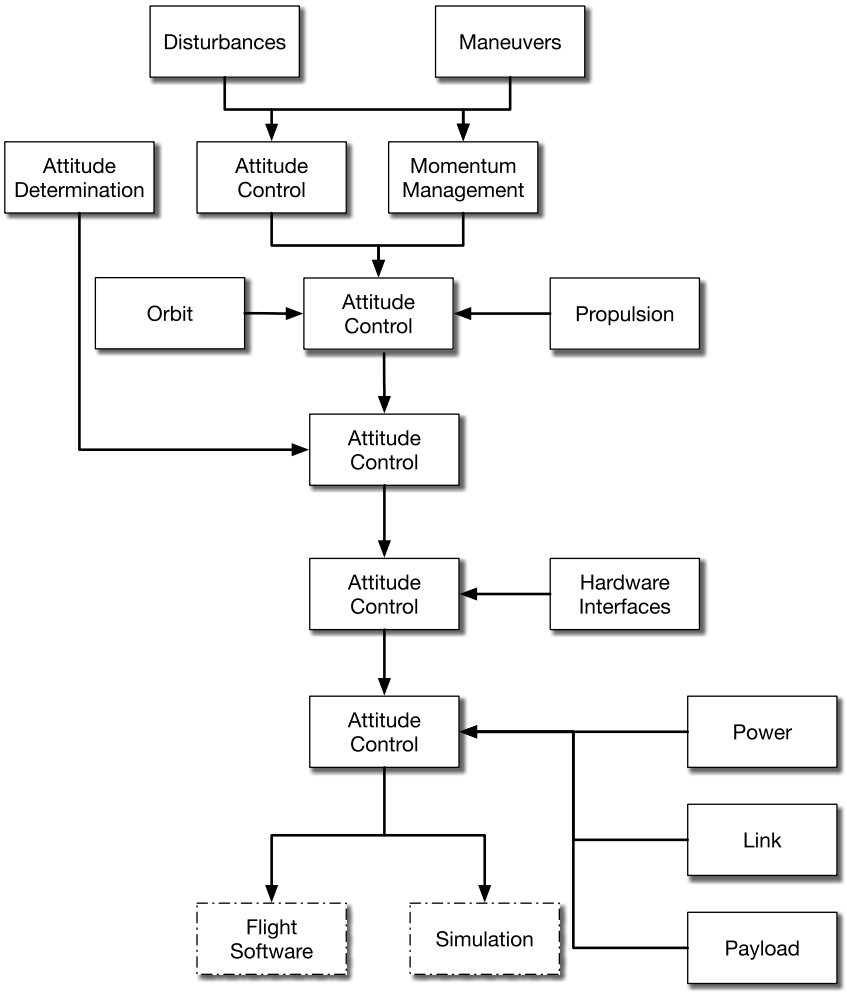


Figure 16.4 Simulation sequence for ACS development. The attitude control block evolves and gains complexity during the design process.

The next step is to add the attitude determination system. Now, the control system is seeing the output of the attitude determination algorithms. Engineers can then easily see the impact of the "real" measurements on the control system.

The next step is to add binary-level interfaces. The interface software converts outgoing torque demands into the required bit patterns for the hardware. Attitude determination receives the simulated words from the sensor. The handling of star cameras is problematic. A full camera model is generally impractical, so a pinhole-camera model is typically used.

The next step is to model the communications, payload, and power systems so that the impact of attitude control on those subsystems. Simulations are used to determine the dynamical effects.

Finally, the flight software is split and loaded onto an equivalent flight processor. The simulation runs on any computer. The hardware models in the simulation should be connected to the same network as on the spacecraft. At this point, the performance of the attitude control system can be assessed. Other subsystem software, for power, payload, and communications, will also be added so that a full flight-software model can be tested.

16.4.2 Analysis support

The simulations in this book are examples of simulations for analysis support. The simulation is used to verify analysis results. This is a good way of catching errors. In many cases, you would write custom code just to verify your analysis results. Since these short simulations usually run quickly, you can do Monte Carlo studies. Monte Carlo studies are running large numbers of simulations with random initial conditions and parameters. When using such simulations they need to be documented and the results recorded to help you study results from more complex or higher-fidelity simulations. One advantage of such simulations is that they can be analytically verified. Some examples are given in the following sections.

16.4.2.1 Sizing of a reaction wheel

Assume the reaction wheel is sized for a particular maneuver. A simulation is built and run to verify the momentum and torque requirements. Complex attitude profiles can be simulated, including profiles that may not be in the baseline operations.

16.4.2.2 Disturbance modeling

Attitude disturbances are computed for a given orbit and attitude profile. A simulation could be run to verify the numbers. If a Fourier series was generated for the profile and orbit it could be compared with the disturbance model itself.

16.4.2.3 Control design

Step and impulse responses could be generated for the controller to verify that the gains get the desired response. Actuator limits and the response to those limits can be tested. The effect of noise filters can be verified by adding noise of varying standard deviations and spectrum to the inputs. Single-axis simulations are a good starting point. These can be followed by rigid-body simulations. If reaction wheels are used, running with and without friction is a good idea.

For thruster systems, you can first look at the ideal response to your thruster control system. You can then add the thruster minimum pulsewidth, pulse resolution, and impulse versus pulsewidth models.

16.4.2.4 End-to-end testing

End-to-end testing means testing the ACS completely as it would be used in orbit. This means

1. Processing commands;
2. Running ACS;
3. Observing the results in telemetry.

Fig. 16.5 shows the flow. Even a fully autonomous system is initiated from the ground, or perhaps a switch on separation. Command processing takes data from the command stream and converts it into engineering units for the ACS. If a sequence of commands has been uploaded, it provides the timing mechanism for the commands. This type of simulation will catch many problems. For example, as part of the design, ACS engineers will specify what parameters can be uploaded as commands. It may be necessary to upload parameters, or other commands in a specific order. End-to-end testing can determine what happens if this is not the case, or catch unanticipated problems with updated control-system parameters.

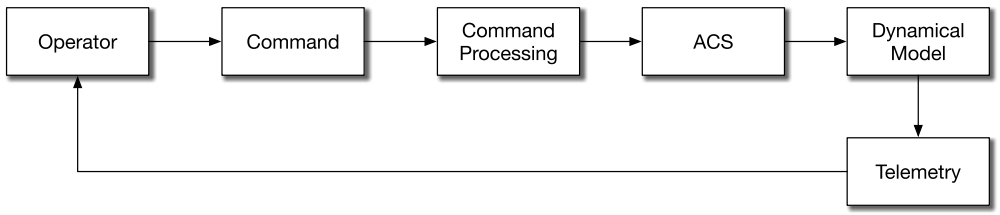


Figure 16.5 End-to-end testing involves the entire operational chain.

An end-to-end simulation can be used as part of operator training, as discussed in Section 16.4.5. Since operators deal with all of the spacecraft subsystems, additional models are usually added.

16.4.3 Performance verification

16.4.3.1 Single case

When debugging a control system it is good to start with the same conditions for each test. This means the same initial states and set the random-number generators to the same initial value. This makes it easy to find problems and compare results for different control-system gains, etc.

16.4.3.2 Grid test

A grid test is what is done after the single case stage is complete. The grid is a small set of initial states and parameters. The initial states should be representative of the conditions expected in flight. The simulation runs for those cases. If a particular case causes a problem, then it can be repeatably run until the problem is solved. Each time a problem is found, the initial conditions should be saved for future testing. The grid test includes variations in parameters that are uncertain. This can include noise levels for sensors, friction levels for reaction wheels, and impulse bits for thrusters. For hardware that has multiple copies on the spacecraft, it is a good idea to vary the parameters among the devices.

16.4.3.3 Numerical gain and phase margins

Typically, a control system is required to meet gain and phase margins. The gain margin is how much the forward gain can change before the system goes unstable. The phase margin is how much the phase can change before the system goes unstable. The phase margin is related to the delay margin in which the quantity of interest is how large a delay the system can tolerate. Some control systems, such as the phase-plane systems used on the Space Shuttle Orbiter [2], are not designed using linear control techniques and are not designed with specific gain and phase margins.

There are several ways to compute margins. One is to use describing functions [3]. The describing-function approach is shown in Example 16.4. A simulation of the phase-plane controller is first run to show that it works very well. The thick green line shows the optimal trajectory. It tracks downward in the phase plane until the first switching line is reached. It then coasts horizontally and finally follows the second switching line to the origin. A very small time step is used to prevent overshoot.

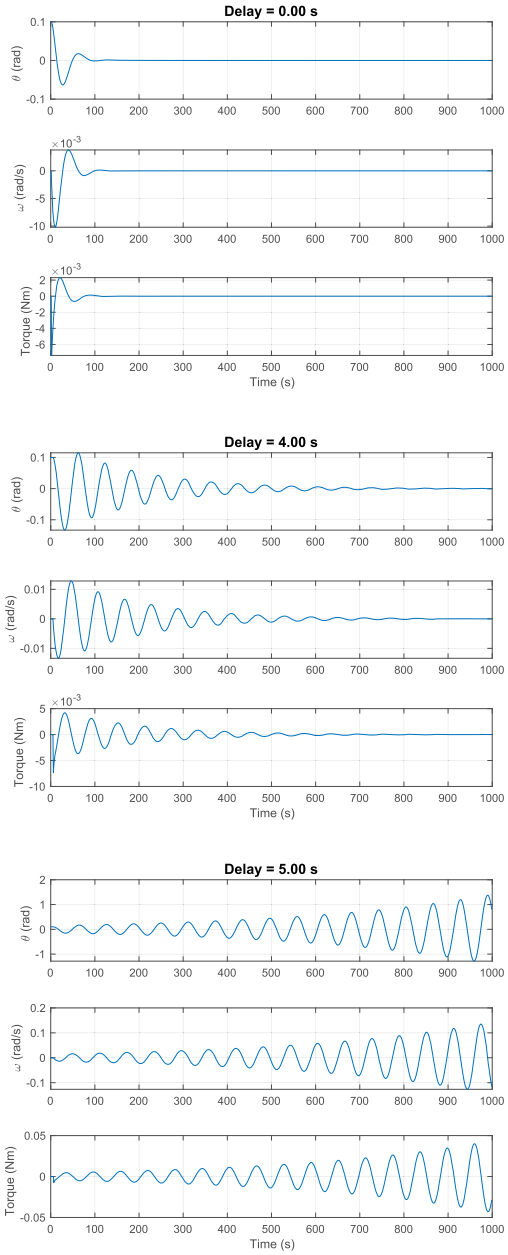
Another method is to vary the phase and gain of signals entering the controller within the simulation. Gain variation is straightforward. Phase variation can be done by adding delays or by adding a phase lag in the control loop. For delays that are a fraction of a control period, the simulation needs to run faster than the control period. Example 16.3 shows delays that are multiples of the control period. A five-second delay makes the system unstable.

This can be applied to a nonlinear control system. Example 16.4 shows a phase-plane controller with and without a delay. The second simulation is run twice as long to show the limit cycle at the origin. The delay causes it to miss the switching lines.

16.4.3.4 Monte Carlo

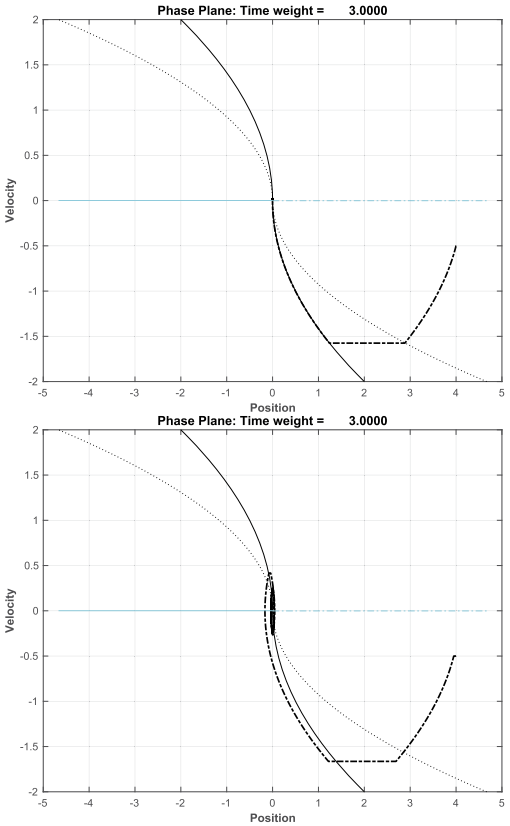
Monte Carlo simulation is a verification method that uses random inputs. Monte Carlo simulation is named after the well-known casino town in Monaco [4]. Rather than use a

```
n               = 1000;
xP              = zeros(3,n);
dT              = 1;
xPID            = [0;0];
x               = [0.1;0];
dRHS            = struct('inr',3,'torque',0);
nDelay          = 6;

[a, b, c, d]    = PIDMIMO( dRHS.inr, 0.7071, 0.1,
        40, 0.1, dT );
tD              = zeros(1,nDelay);

for k = 1:n
  xPID              =  a*xPID + b*x(1);
  xP(:,k)           = [x;dRHS.torque];
        dRHS.torque       = tD(1);
  torque            = -c*xPID - d*x(1);
        tD(1:nDelay-1)  = tD(2:nDelay);
        tD(nDelay)      = torque;
        x               = RK4(@RHS,x,dT,0,dRHS);
end

t  = (0:n-1)*dT;
yL = {'\theta (rad)','\omega (rad/s)','Torque (Nm)'};
s  = sprintf('Delay = %4.2f s',(nDelay-1)*dT);
Plot2D(t,xP,'Time (s)',yL,s);

%% Right hand side [angl;ang2;omegal;omega2]
function xDot = RHS(x,~,d)

xDot      = [x(2);d.torque/d.inr];

end
```



Example 16.3: Control response with different delays. With a 5-second delay, the system is unstable.

```
n               = 8000;
xP              = zeros(3,n);
dT              = 0.001;
x               = [4;-0.5];
dRHS            = struct('inr',3,'u',0);
nDelay          = 100; % 1 for no delay
tD              = zeros(1,nDelay);
kOpt            = 3;

for k = 1:n
  u               = PhasePlane(x(2),x(1),kOpt
        ,0.001,0.001);
  xP(:,k)         = [x;u];
        dRHS.u          = tD(1);
        tD(1:nDelay-1)  = tD(2:nDelay);
        tD(nDelay)      = u;
        x               = RK4(@RHS,x,dT,0,dRHS);
end

PhasePlanePlot( xP(2,:), xP(1,:), kOpt, 0.001,
      0.001, [-2,2] )

%% Right hand side [ang1;ang2;omega1;omega2]
function xDot = RHS(x,~,d)

xDot    = [x(2);d.u];

end
```



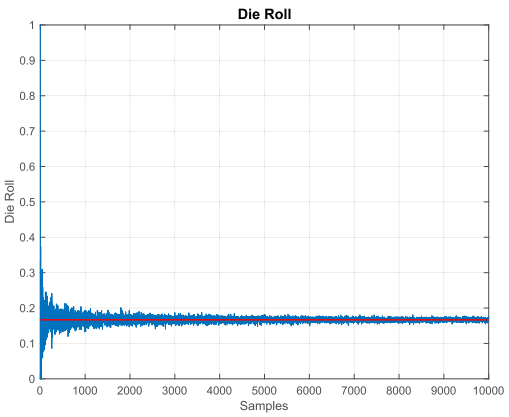
Example 16.4: Phase-plane controller. The first plot shows the phase-plane trajectory in green without delay. The switching lines are also shown. The second shows it with the delay.

deterministic set of inputs, the initial conditions and model parameters are chosen from a probability distribution. Besides possibly catching errors, it allows you to do sensitivity analyses and determine the correlation of inputs.

It is interesting to get a feel for the number of cases needed for a Monte Carlo simulation process. Example 16.5 gives the example of a single dice. The probability of any number is 1/6. Converging to that value takes many samples.

Even for this simple case, nearly a thousand tests are needed to converge on the known probability of a given digit resulting from the throw of the dice. The idea of the Monte Carlo simulation is to pick random samples that cover the range of possible initial conditions and parameters without having a computationally impossible number of cases.

```
kMax = 10000;
a = zeros(1,kMax);
for k = 1:kMax
  r      = randi(6,1,k);
  j      = find(r==1);
  a(k)   = length(j)/k;
end

Plot2D(1:kMax,a,'Samples','Die Roll')
line([1 kMax],[1/6 1/6],...
  'linewidth',2,'color',[1 0 0]);
```



Example 16.5: Die roll. The probability that a dice roll will be a given number is 1/6. It takes many trials to converge to that value.

16.4.3.5 Commands

Most ACS are commandable. You can upload new gains and send commands to actuators using commands from the ground. These need to be exercised. Commands must be shown to mesh with the other ACS operations. Incorrect sequences of commands, such as turning on a hydrazine thruster before turning on the catalyst-bed heater, should produce the correct response.

16.4.3.6 Edge cases and stress cases

Edge and stress cases are when the sensors and actuators are operating at their operational limits. For example, a spacecraft can be given a maneuver with the starting value of the reaction-wheel inertia near its maximum value. Another case is when a star sensor enters the Sun or Earth stay-out zone. Operations that transfer from one sensor to another should be tested. If your spacecraft has attitude control modes, transitions between modes should be simulated. For a spacecraft that is entering a planetary atmosphere, the maximum allowable entry conditions should be exceeded in testing. This is especially important for landers on other planets or moons where the atmosphere model will not be as well known as the Earth's.

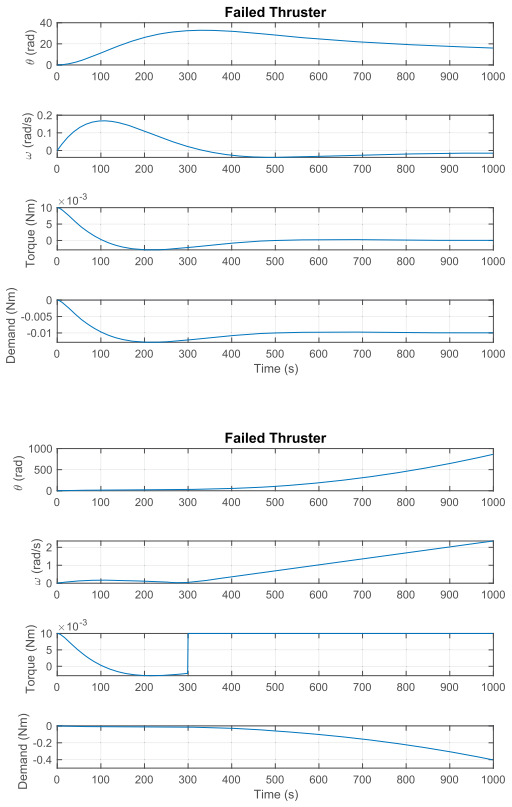
16.4.3.7 Failure cases

Failure cases need to be simulated to understand the dynamics of a failure. For example, a pointing-control system that uses an Earth sensor will not point correctly if the Earth sensor fails. The response to a failure may result in operational problems beyond the control system not pointing correctly. To simulate a failure it is necessary to understand

the failure modes of the devices. For example, thruster valves are generally designed to fail closed as failing open would result in a continuous torque on the spacecraft.

Example 16.6 shows an example of a spacecraft with two thrusters. It has a PID control law. When one thruster fails, the torque can only have one sign. In this case, the disturbance is positive so a failed positive failed sign is unobservable. If the failure sign is negative, the attitude diverges because the total torque is just the disturbance. As this shows, it is necessary to test the response for all thruster failures, and also to run simulations with different disturbances and initial conditions.

```
% Failed thruster simulation

n          = 1000;
xP         = zeros(4,n);
dT         = 1;
kFail      = 300;
failSign   = 1;

torque     = 0;
inr        = 3;
xPID       = [0;0];

x          = [0;0];
dist       = 0.01;

[a, b, c, d] = PIDMIMO( inr, 0.7071, 0.01, 1000,
       0.1, dT );

for k = 1:n

  % Controller
  xPID     =  a*xPID + b*x(1);
  torqueD = -c*xPID - d*x(1);

  % Simulate failure
  torque  = torqueD;
  if( k > kFail )
    if( sign(torque) == failSign )
      torque = 0;
    end
  end
  xP(:,k) = [x;torque + dist;torqueD];
  x       = RK4(@RHS,x,dT,0,inr,torque+dist);
end

t  = (0:n-1)*dT;
yL = {'\theta (rad)', '\omega (rad/s)' 'Torque (
       Nm)' 'Demand (Nm)'};
Plot2D(t,xP,'Time (s)',yL,'Failed Thruster');

%% Right hand side
function xDot = RHS(x,~,inr,torque)

xDot = [x(2);torque/inr];

end
```



Example 16.6: Failed thruster. When the torque demand is the same as the failure sign, no control torque is produced.

16.4.4 Interface verification

Interface verification is used to make sure that the flight computer can communicate with the sensors and actuators on the spacecraft. Fig. 16.6 shows a general interface.

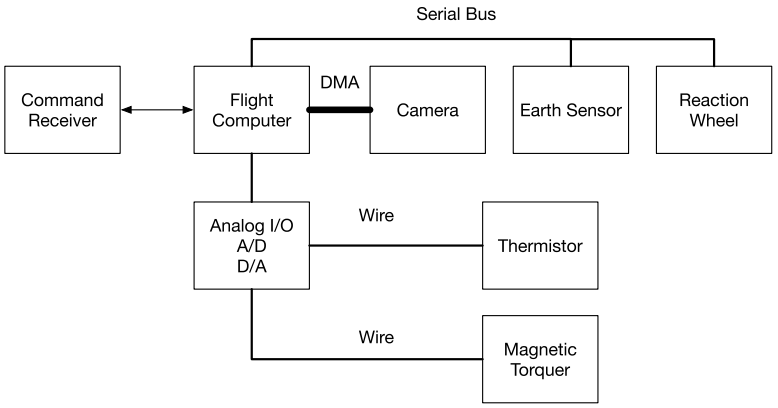


Figure 16.6 Hardware interfaces showing interface types. Direct memory access (DMA) is typically used for cameras although modern (terrestrial) serial interfaces are very fast.

Interfaces are generally through a serial data bus, like RS-422, SpaceWire, or MIL-STD 1553, or through single wires. Single wires provide analog signals. They may go through an analog-to-digital converter when coming from the sensor or a digital-to-analog converter when going to an actuator. For on/off signals a voltage pulse may be sufficient. Often the analog I/O is on a separate board. Modern microcontrollers may have multiple I/O pins on the motherboard. DMA is direct memory access and is useful if a sensor is transferring a large amount of data, like an image, to the flight computer.

It is rarely necessary to run a dynamical simulation to perform interface verification. If your flight software allows for commands, this is a clean way to exercise the interfaces.

16.4.5 Operator training

Operator training requires an interface much like the one shown in Fig. 16.5. Since the operators look at more than just ACS sensors, it is necessary to simulate the power, payload, communications, and thermal subsystem. Operators work from procedures. Once the procedures have been established, operators will work through them using the training simulation. It is not always necessary to have a dynamic simulation. In many cases, it is sufficient for the telemetry to just indicate that a command has been sent.

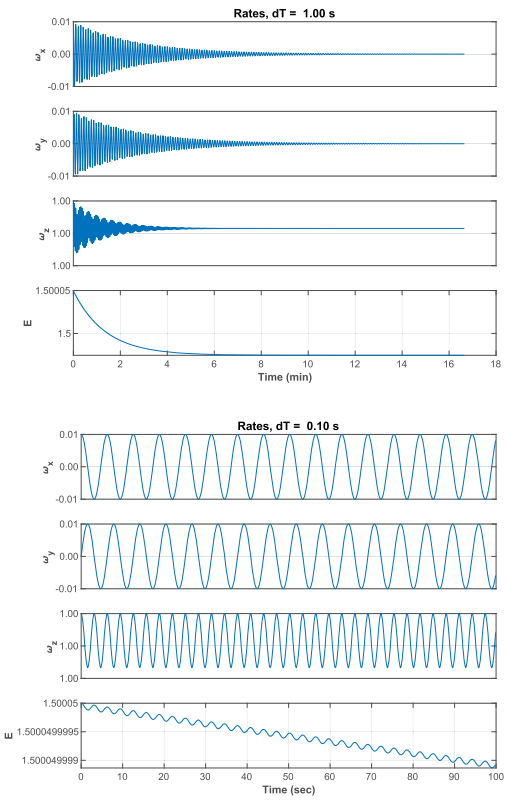
Fancier simulations are like computer games in which anomalies can be injected by a trainer. This was standard during the Apollo and Shuttle eras and presumably is used in the modern SpaceX and Boeing human-carrying spacecraft. For spacecraft with crews, the crew member is the operator. These may also involve operators so they can get quite complex. If a dynamical simulation is used, it is good to use the same one that was developed for the ACS. New simulation models may be necessary for the thermal and power subsystems as those subsystems do not always concern themselves with thermal- or power-system dynamics.

16.4.6 Anomaly investigations

An anomaly may involve an ACS not performing as expected or a complete disaster like Mars Observer vanishing during its orbit insertion burn. In any situation, simulations can be part of the anomaly investigation. If an end-to-end simulation has been built then it can be used. In many cases, the smaller specialized simulations used for ACS development may be more useful. For example, if the star camera seems to be misbehaving the attitude determination simulations may be valuable. When doing an anomaly investigation it is important to be certain that your models can represent the issues that seem to be causing the problem. For example, the Hubble mirror anomaly clearly could not be studied by a pinhole-camera model of the optical system.

Before using the simulation tools it is important to be sure they model the spacecraft as observed onorbit. This can be difficult because telemetry data is limited. The sampling rate is usually low and the resolution of the data is also low. You may not be able to observe high-frequency effects since the sampling rates and resolution of the data are

```
%% Nutation

timeStep = [1  0.1];
n        = 1000;
inr      = diag([1;2;3]);

for j = 1:2

  xP  = zeros(8,n);

  dT = timeStep(j);
  x = [1;0;0;0;0.01;0;1];
  for k = 1:n
    omega    = x(5:7);
    e        = 0.5*omega'*(inr*omega);
    xP(:,k) = [x;e];
    x        = RK4(@RHS,x,dT,0,inr);
  end

  t = (0:n-1)*dT;
  yL = {'q_s' 'q_x' 'q_y' 'q_z'...
  '\omega_x' '\omega_y' '\omega_z' 'E' };

  s = sprintf('Rates, dT = %4.2f s',dT);
  TimeHistory(t,xP(5:8,:),yL(5:8),s);

end

%% Dynamics right-hand-side
function xDot = RHS(x,~,inr)
q        = x(1:4);
omega    = x(5:7);
omegaDot = -inr\cross(omega,inr*omega);
xDot     = [QITOBDot(q,omega);omegaDot];
end
```



Example 16.7: Artificial damping. The spacecraft appears to have damping with a time step of one second. The second simulation has the proper nutation response.

too low. Some spacecraft allow you to selectively download data at higher sample rates. This can be useful. If you have end-to-end simulations that produce simulated telemetry, these can help debug on-orbit issues.

16.5. Artificial damping

Simulations, because they are approximations, can give surprising results. Example 16.7 is a simulation of a spinning rigid body without any damping. The simulation is run at two time steps. At the larger time step, it appears that the angular rates damp. This is impossible and is entirely an artifact of the time step being too large.

Care needs to be taken with simulations to make certain that numerical artifacts are not misinterpreted as valid results.

References

[1] D. Hough, IEEE 754-2019 - IEEE Standard for Floating-Point Arithmetic, Tech. Rep., IEEE Computer Society, July 2019.
[2] M. Paluszek, The dynamics of the space shuttle orbiter with a flexible payload, Acta Astronautica 10 (4) (1983) 163–178.
[3] M. Paluszek, Analysis of digital controllers using frequency-dependent describing functions, IEEE Transactions on Automatic Control 32 (7) (1987) 651–653.
[4] Anonymous, Monte Carlo simulation, https://www.ibm.com/cloud/learn/monte-carlo-simulation, August 2020.

CHAPTER 17

Testing

17.1. Space story

The first use of the Improved Electrothermal Hydrazine Thrusters (IMPEHTs) was on an operational mission. These thrusters had higher exhaust velocities due to a more powerful heater. As a consequence, the propellant feed tubes were narrower. The IMPEHTs could be off-pulsed. Thus the four thrusters on the North face of the spacecraft could be used for both North/South station keeping and attitude control during the burn. Off-pulsing was standard practice for East/West maneuvers that used standard hydrazine thrusters. This eliminated the need for hydrazine thrusters for roll/yaw attitude control. When the first maneuver started, the control performance was not as expected. It should have been very smooth. Instead, the spacecraft wobbled about. It was controlled but the attitude motion was greater than expected.

This launched an anomaly investigation. One path was to verify that the valves were behaving as expected. This test was done with a similar spacecraft in a high-bay. We did the test using PCs with LabView, the first time this had ever been done. LabView would send commands to the valves. The results were that the valves behaved as expected. Ultimately, it was determined that the problem was helium bubbles in the fuel. The new feed tubes were so narrow that bubbles could block the tube and hot helium did not produce any thrust. The bubbles were introduced to the fuel during pressurization. The propulsion system was a blowdown system that used helium as the pressurant.

17.2. A testing methodology

This section provides a discussion of testing of an attitude control system (ACS) and presents a test program necessary to ensure that the probability of failure of the attitude determination and control subsystem (ADACS) software due to hidden bugs is sufficiently low. This section also covers the testing of the flight software test-bed and supporting tools.

17.3. Reliability

The ACS must be able to meet the requirements for the spacecraft. That is, the system must be able to operate and meet all pointing requirements for the life of the spacecraft. Reliability will be a function of the reliability of the hardware and software. Hardware reliability is tied to the mean time between failures (MTBF). Most hardware follows a

https://doi.org/10.1016/B978-0-32-399915-1.00028-0

bathtub curve for its probability of failure. The reliability is [1]

$$R(t) = e^{-\alpha t^{\beta}} \tag{17.1}$$

where

1. $\beta < 1$ Infant mortality;
2. $\beta = 1$ Useful life;
3. $\beta > 1$ End-of-life.

The failure rate is

$$Z(t) = \alpha\beta t^{\beta-1} \tag{17.2}$$

where α is the failure rate at $t = 0$. β changes, typically, at discrete times. To find the new α for the next segment, when β is different

$$\alpha(t(j)) = \frac{Z(j-1)}{\beta(t(j))t(j)^{\beta(t(j)-1)}} \tag{17.3}$$

A typical bathtub curve is shown in Fig. 17.1. This type of curve applies to the hardware

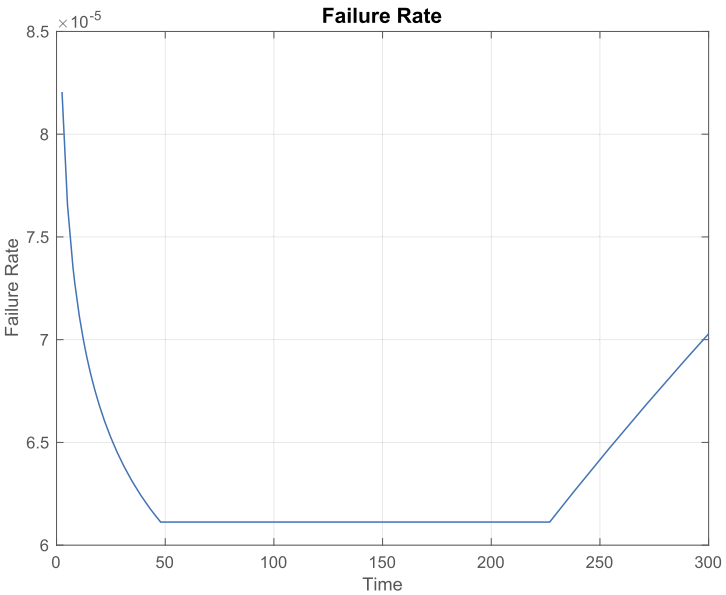


Figure 17.1 Bathtub curve. The end-of-life curve is shaped differently from the beginning-of-life curve.

in the ACS system. For parts in series the reliability is [2]

$$r = \prod_{k=1}^{N} R_k \tag{17.4}$$

For parts in parallel for a system where r must survive, where all Rs are equal

$$R = \sum_{x=r}^{N} \binom{N}{x} R^x (1-R)^{N-x} \tag{17.5}$$

Infant mortality is dealt with by preflight testing. The flight portion is the bottom of the bathtub. The end-of-life is unavoidable. The reliability of the components must be sufficient that the system does not fail before its required lifetime.

17.3.1 Requirements flow and testing

Fig. 17.2 shows the requirements flow for flight software. The figure shows only the requirements levied by other subsystems that the flight software must support, such as the ADACS, power subsystem, and thermal subsystem. Requirements levied directly on the flight software are not shown.

Each subsystem has performance requirements that can only be met through operation of the flight software. This includes closed-loop attitude control, power management, etc. During the design cycle, each subsystem team performs analyses and simulations to demonstrate that the subsystem design meets its requirements. The subsystem then generates software requirements and these are realized in the flight software. Flight-software testing verifies that it meets the requirements levied on it by the subsystems, but not necessarily that the subsystem design, as implemented in flight software, meets the original subsystem requirements. In practice, this usually happens indirectly since flight software is often verified by comparing simulation results with the flight software to simulation results generated by subsystem simulations that have been shown to meet the performance requirement. The validity of this indirect approach is based on the assumption that the test platform for the flight software has sufficient fidelity to make the comparisons (in terms of performance) meaningful. In addition, the subset of tests run on the flight-software test platform are assumed to be representative of the subsystem tests originally used to demonstrate that the subsystem met its requirements. These assumptions may not be valid if the flight-software test platform simulation is of insufficient fidelity.

A careful distinction must be made among the types of requirements. Each subsystem has requirements it must fulfil and many fulfil their requirements through flight software. They do this by levying requirements on the flight software. In theory, if the flight software meets these requirements levied upon it by the subsystems, the subsystems will also meet their requirements. Unfortunately, this may not be the case and it is usually necessary to exercise the actual flight software and a subset of the subsystem hardware to demonstrate that the subsystem requirements have been met. Furthermore, some iteration between the flight-software-design phase and the subsystem-design phase is almost always necessary since subsystem requirements may not be implementable as first specified.

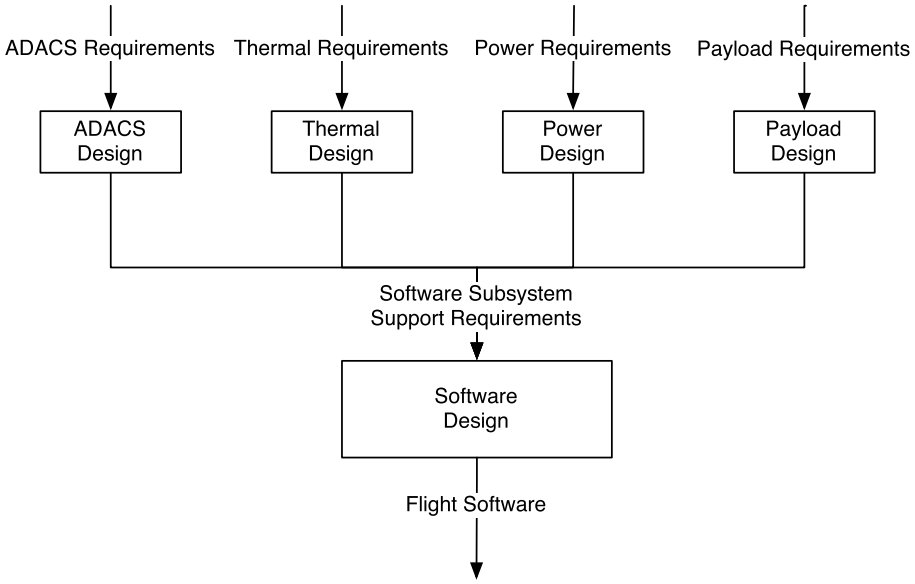


Figure 17.2 Typical requirements flow for an attitude control system.

17.3.2 Testing lifecycle for the ACS flight software

An example of a complete testing lifecycle for a flight-software product is given in Fig. 17.3. The rectangular symbols represent the design phases and the boxes with the rounded corners represent test methods or phases that are used to validate the results of a design phase before proceeding to the next phase. The first two phases in the design lifecycle involve the design of the ADACS algorithms, as opposed to the software that implements them in the flight computer. This phase is driven by the ADACS requirements. In the first phase, the ADACS algorithms are designed and tested in a control-analysis package. The design is then validated using a high-fidelity spacecraft simulation. In this case, the sensor, actuator, environment, and dynamics model are high fidelity. Flight computer arithmetic and interfaces are not modeled. The output of this phase is the ADACS specification and requirements documents along with supporting analysis memos.

The flight-software ADACS support requirements are derived from the results of the ADACS analysis. The remaining steps follow a standard software-design cycle. The test tools and platforms shown in the figure will be discussed in more detail later in this chapter.

The validation matrix lists requirements and the software modules that satisfy the requirements. A code walkthrough involves reading through the source code. Traditionally this is done by reading through a print out of the code or reading it in a text

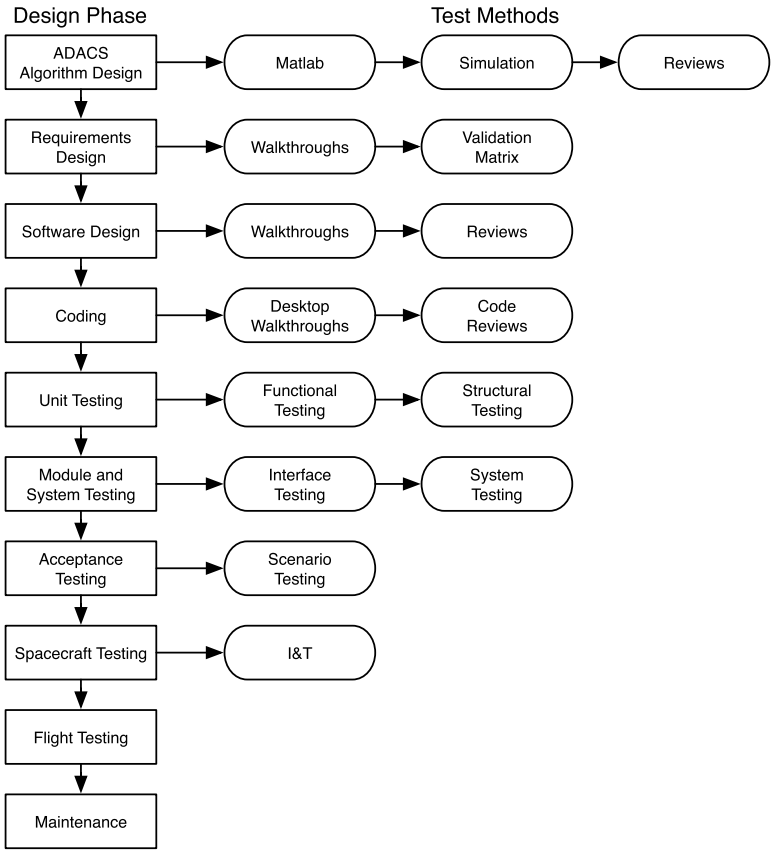


Figure 17.3 Testing lifecycle from algorithm design through flight.

editor. A variant is to walk through the code in a source-code debugger so that the flow of execution can be studied.

Integration and test (I&T) is where the spacecraft is assembled. The word "integration" is used because this stage involves the assembly of manufactured parts.

17.4. Flight-vehicle control-system testing

Flight-vehicle control-system testing goes through two stages. These are defined in Table 17.1. The actual flight vehicle is rarely available for test purposes until late in a program. Due to scheduling problems, it is rarely possible to perform extensive closed-loop testing on the spacecraft when it is in integration and test. Testing on the complete spacecraft is limited by the lack of internal visibility into the software. The only access to the flight software is usually through the command and telemetry interface. It is

difficult to gain access to internal variables in the flight software, making debugging difficult. Sensor inputs must be generated either by external targets, that provide little flexibility, or by inserting inputs into the lines running from the sensors to the flight computer. Measurements of the outputs are limited by whatever measurement channels are available on the actuators. For example, if the only speed measurement on a reaction wheel is the tachometer, it will not be possible to know the true reaction-wheel state. Mission-orbit operations, such as station keeping, are tested by performing the shortest

Table 17.1 Flight-vehicle-software testing.

Stage	Description
Preflight	All functions are tested either through closed-loop simulation or with open-loop test data (usually generated by another closed-loop simulation). Simulations may include some, none or all of the ADACS hardware. ADACS algorithms may be implemented in an abstract form, in another language (such as FORTRAN) in the target language, C, but on a different processor, and in the target language on the flight processor.
Flight	Transfer-orbit and acquisition ACS functions are tested as each is introduced in the mission sequence. Problems are usually resolved through procedural changes and work arounds. If necessary, the flight software can be patched.

possible version of the operation. Problems are usually resolved through procedural changes and work around. If necessary the flight software can be patched.

Simulating the complete space environment, including thermal, vacuum, and dynamics effects, is impractical with a complete spacecraft. In addition, it is difficult to simulate offnominal or worst-case conditions involving the actuators and sensors.

The most cost-effective solution is to run a simulation of the spacecraft dynamics and environment on another computer and use taps on the sensors and actuators to close the loop. This provides the most complete picture of how the flight software will perform, but must be used in conjunction with a high-visibility tool, such as a development station or processor emulator, to help trace problems that arise. Flight-software designers must rely on simulations of the vehicle and the operational environment to validate the flight software. Simulations are of five types, as given in Table 17.2.

Each of the simulations may include a complete closed-loop dynamics model of the spacecraft or may be used open loop with test data from another source usually from another simulation. Industry experience with different ADACS simulation testbeds is given in the next section.

17.5. Test levels (preflight)

This book uses the three levels of testing for the flight software. Unit testing is usually done by the designer of the procedure. It is meant to be exhaustive and guarantees

Table 17.2 Simulation types used to verify flight software.

Simulation	Description and Use
All software.	The vehicle, flight computer, and interfaces are modeled in software. This makes it a very flexible and portable tool. The fidelity of the simulation is limited by many factors including: **1.** Available component data; **2.** Host computer throughput; **3.** Time available to design, build, and test numerical models. It is assumed that the fidelity of the simulation is sufficient to verify the performance of the algorithms in the flight software. Since the simulation can usually run many times faster than real-time, it is possible to test the long-duration behavior of the algorithms.
All software with flight-computer emulation.	The flight computer is emulated. This provides more accurate timing information at the expense of execution speed. It also gives excellent visibility into the workings of the software. Emulation can uncover compiler bugs or instruction-set problems.
All software with a flight computer in the loop.	This permits real-time operation. If dual-port memory is available the inner workings of the flight software can be monitored. However, if flight-prototype components are used, the visibility into the flight software is usually limited by the telemetry interface.
Software dynamics, all ADACS hardware in the loop but fixed base.	All interfaces are faithfully represented. The closed-loop simulation generates the stimuli for the sensors so sensor dynamics are not tested. This is sometimes done with a hardware breadboard ACS system. It can also be done on the actual spacecraft.
Software dynamics, all ACS hardware in the loop with a moving base.	All of the sensors are stimulated by sources representative of actual operation. For spacecraft with gyros this requires a multiaxis rate table. Earth and Sun targets are needed for the Earth and Sun sensors. Orientation of the rate table and Sun/Earth targets must be synchronized. This provides the highest possible fidelity, short of using the flight vehicle in operation. The cost is usually prohibitive for all but the most expensive satellites.

that the procedure will work correctly for all valid inputs. Since it is rarely possible to check every valid input, the inputs must at least encompass the full range of valid inputs. Boundary values must always be included in these tests. A combination of boundary values, random inputs (within the valid ranges), and "best guesses" of inputs that may cause problems, should be employed. Random inputs should be normally distributed about the mean input values. In addition, all logic paths must be exercised. The former type of testing is called black-box testing, the latter, in which test inputs are chosen to follow all logic paths, is known as white-box testing. Whenever possible, the procedure is tested against analytical results or, if that is not possible, against already verified, higher-fidelity models. Unit testing must be exhaustive because it is not feasible to fully test a procedure at the higher testing levels since it is difficult to guarantee, or prove, that

a given routine has been fully exercised. Unit testing is usually open loop, although closed-loop test drivers may be employed, see Table 17.3.

Table 17.3 Test levels.

Level	Scope	Tests
Unit	Individual procedures	Range of valid inputs. All paths and all code.
Module	Individual functions	All mission operations relevant to that module.
System	Entire ADACS	All mission sequences.

Module testing tests functions of the flight software. For example, the backup ephemeris is a function and is implemented by means of several different procedures and/or packages in the flight software. Module testing demonstrates that independent functions of the software work properly and that all of the procedures that comprise a module interact correctly. Module testing is usually open loop, although closed-loop test drivers may be used.

Module testing is the first step in verifying the interfaces between procedures. This is also accomplished during system testing. For some modules, such as thruster commanding, module testing would include testing of the hardware interfaces.

Most of the lower-level flight-software requirements will be validated using either unit or module-level testing, if only because the requirements are partitioned by function, and sometimes by module. Module testing should demonstrate that the module will meet its requirements over the full range of operating conditions. While unit testing should include all valid inputs, module testing need only include valid inputs within the operational range.

The highest level of testing, system testing, demonstrates that all of the modules work together correctly and that the flight software can perform all required mission operations. System testing should include realistic command inputs through the appropriate hardware.

All tests should employ a coverage-monitor tool. A coverage-monitor tool keeps track of what code is exercised in each test. For a given module, it is essential that all code be exercised in its module test. At the system level, all code should be exercised in at least one of the system-level tests. Studies have shown that testing done without coverage tools typically exercises only 55% of the code. In reality, it is impossible to actually test every path in anything but the simplest software and this must be factored into the software-development and quality-assurance processes.

17.6. Test levels (flight)

The first flight tests of the ADACS software will happen during actual mission operations in transfer orbit. Each function can be tested when it is first used. Whenever

possible, the duration of use of the function should be limited. Many spacecraft are first put into safe-mode prior to undergoing onorbit testing. This is rarely necessary if the software has been properly tested prior to flight. In many spacecraft it is necessary to precede immediately into transfer-orbit operations, so the safe-mode approach will not work. Consequently, it is desirable to develop a mission sequence that keeps the spacecraft progressing through mission phases without unnecessary risk to the spacecraft or mission.

17.7. Simulations

Simulations are a critical element in developing a spacecraft. It is not possible to test a spacecraft in the operational environment so simulations are required. For ADCS purposes the minimum simulation-state equations are

$$\dot{\omega} = I^{-1}\left(T - \omega^{\times} I\omega\right) \tag{17.6}$$

$$\dot{q} = f(\omega, q) \tag{17.7}$$

$$\dot{v} = F - \mu\frac{r}{|r|^3} \tag{17.8}$$

$$\dot{r} = v \tag{17.9}$$

where F and T are functions of external disturbances and external controls such as thrusters and magnetic torquers. The external disturbances will be a function of r and v thus it is important to simulate the orbital dynamics along with the attitude dynamics. If the spacecraft were a gyrostat, additional equations for the reaction wheels would be needed. This dynamical model is sufficient for design of an ADCS of a rigid-body spacecraft.

When you first build a simulation, use ideal sensor and actuator models. This allows you to verify your control laws. The next step is to accurately model sensors and actuators and test with the detailed models to make certain the performance does not change and to identify any issues due to the sensors and actuators.

Occasionally, moving-base simulators and air-bearing simulators are used. While sometimes Sun and sky targets are used, often the sensors have an input that allows you to produce an output synchronized with the simulation. Given the sophistication of software models, hardware simulations are rarely needed, except to verify interfaces. This is usually done routinely during integration and test.

For operations and training, large-scale simulations are often developed. It is rarely necessary to use more complex ADCS models, but it becomes necessary to add other subsystems such as power, thermal, and communications. While ACS engineers always work with dynamical models, other subsystems do not. For this reason, ACS engineers are often called upon to help other subsystems add their models to this type of simulation.

17.8. Software-development standards

Many formal standards exist for software development, both commercial and government. Some of the most popular (or most used) are listed in Table 17.4.

Table 17.4 Software-development standards.

Standard	Description
MIL-STD-2167A	Defense system-software development. Covers the acquisition, development, and support of software systems.
NASA-STD-8719.13A	Expands on the requirements of NMI 2410.10, "NASA Software Management Assurance and Engineering Policy," NHB 1700.1(V1), "NASA Safety Policy and Requirements Document"; and NASA-STD-2201-93, "Software Assurance Standard."
CMMI	Capability Maturity Model Integration (CMMI) is an approach to process improvement that is used to set up effective software-development systems.
IEEE	The Institute for Electrical and Electronics Engineers has developed a comprehensive set of software-development standards that are used throughout industry.
UML	Unified Modeling Language (UML) that is implemented using the IBM Rational Rose products provides a structure for development of large software projects from requirements definition through test and evaluation.
ECSS-E-40	European Space Agency standards for software engineering.
ECSS-Q-80	European Space Agency standards for software product assurance.

References

[1] B. Seymour, BMTTF, Failrate, Reliability and Life Testing.
[2] Jeremy Kasidin, Reliability, Princeton University Mechanical and Aerospace Engineering Course Notes..

CHAPTER 18

Spacecraft operations

18.1. Space story

I was at the Charles Stark Draper Laboratory (CSDL) during the launch of the Space Shuttle with the first communications-satellite payload. CSDL designed the Space Shuttle control systems. The communications satellite was mounted on a spin table that would spin it up before separation. It was already in orbit when my boss came to me and asked me to see if the bias momentum would have any effect on the Orbiter control. I ran simulations and the control system worked just fine. In the meantime, my boss did a simple calculation in his head and computed the possible torque from the angular rates of the Orbiter and determined that the $\omega \times h$ torque was insignificant. This shows why having a firm knowledge of physics is so important!

18.2. Introduction

Most people are familiar with images of NASA and JPL mission-control centers. Engineers sit at consoles where they can observe data sent from orbiting satellites. Operators can send commands to the spacecraft. The spacecraft or rovers operated at JPL are robotic. Those operated from Baikunar, NASA/JSC, or NASA/KSFC involve people onboard the spacecraft.

18.3. Preparing for a mission

Preparations for a mission are very similar to preparations for live opera, ballet, or play. A satellite mission is no different from a live theater. Once the performers are on stage they must deal with what happens in real-time. There are no "do-overs." Spacecraft missions are much the same. Preparation for a mission is modeled after theater, as shown in Fig. 18.1.

Fig. 18.2 shows the typical organization for a launch team. In some organizations, this team is permanent and their only job is to operate spacecraft. In others, the subsystem team member joins the launch-support organization only for the launch. Though nominal lines of communication are shown, it is not unusual for subsystem engineers to engage other subsystem engineers when issues arise. The attitude control and mission-planning groups often have the same manager because their functions are tightly interrelated. The Shift Supervisor is usually one of the more experienced subsystem engineers. Working as a shift supervisor is a good way to help develop system engineers.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00029-2

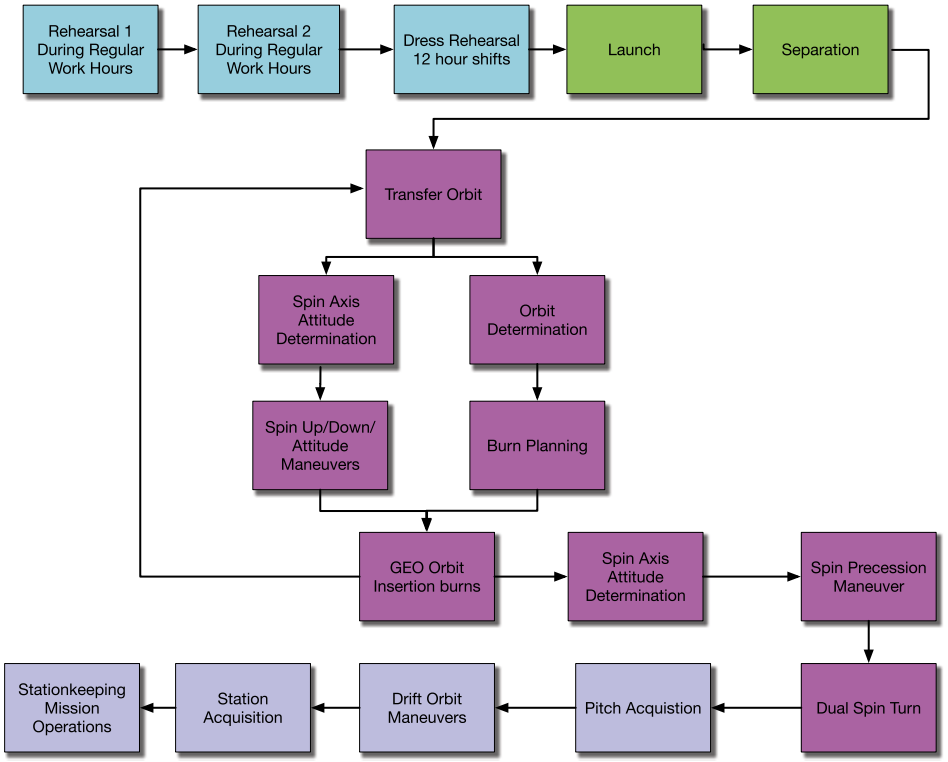


Figure 18.1 Rehearsal schedule for a satellite launch.

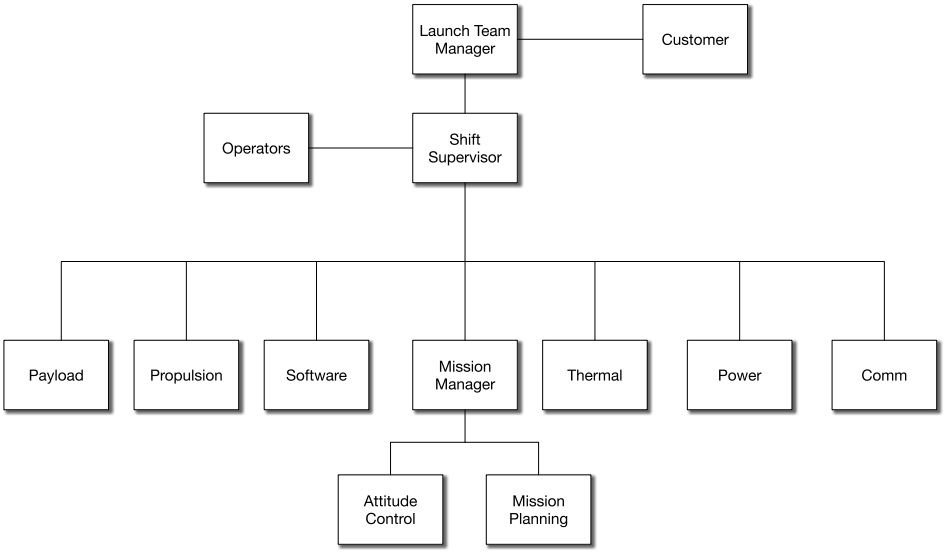


Figure 18.2 Satellite-launch team.

18.4. Elements of flight operations

Elements of mission operations are shown in Fig. 18.3. The satellite is controlled by the flight software and the flight software is commanded via the communication links from the control center. Both the flight and ground software require databases of parameters that include parameters for the flight software, telemetry-frame definitions, command databases, and ground-software parameters. The ground software is used by the flight-operations team to perform analyses, create command sequences, and compute the trajectory and the required orbit changes. The line between the databases and

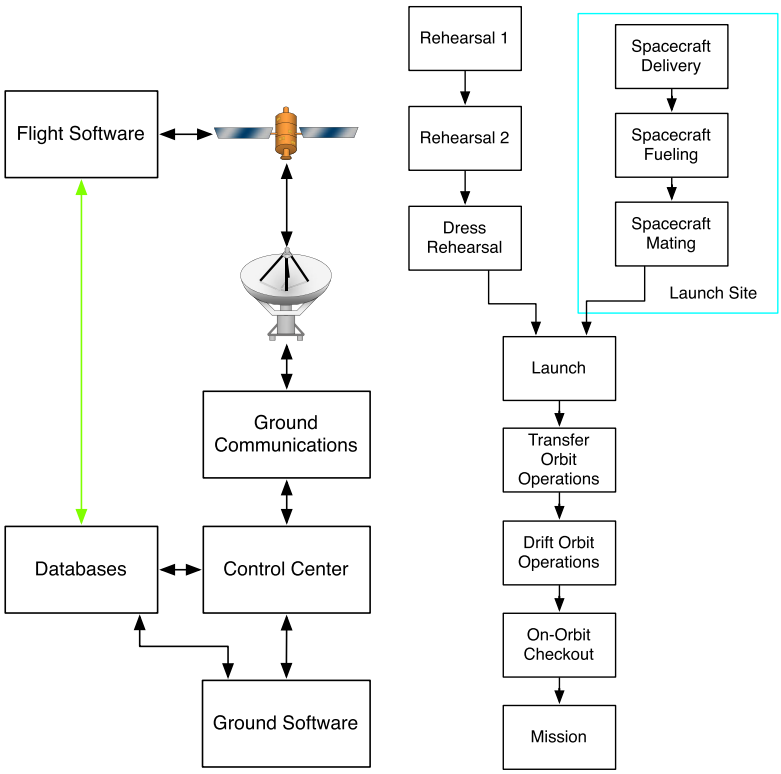


Figure 18.3 Mission-operations elements and timeline.

flight software is virtual. There is no direct link but the databases hold all changeable parameters in the flight software. In some cases, an actual database is a resident on the flight processor but sometimes the parameters are distributed throughout the software itself.

18.5. Mission-operations timeline

The mission-operations timeline is shown in Fig. 18.3. At the end of manufacturing and integration and test, the spacecraft is shipped to the launch site. In parallel, the mission-operations team prepares for the mission operations through a series of rehearsals.

18.6. Mission-operations entities

Many organizations are involved in a satellite launch. An example of the entities and their connectivity is shown in Fig. 18.4. The mission-control center is the center of the launch operations and is supported by several other organizations. The method of connectivity was once the commercial phone system but now is the internet. Management above the level of the operations team may maintain a physical presence. Management only gets involved in the event of serious anomalies. The customer will always have representatives at the control center during the launch but may rely on telecommunications later in the mission. In commercial operations, the customer often has their own control center. NORAD can provide initial orbital elements since they track all satellites.

The operations team communicates with the satellite via ground antennas and the tracking stations. Each tracking station can monitor telemetry and send commands should the phone system go down during critical operations.

Connections to the launch site include the satellite-launch team and the rocket-launch team. The rocket-launch team is in complete control until satellite separation and rarely needs feedback from the satellite-control team.

All entities connect through the communications system. However, not all entities can perform all functions or communicate directly with all other entities. For example, only the control center can send commands to the spacecraft or obtain telemetry. The customer usually cannot talk with the spacecraft until handover occurs.

The data rates of the various links vary greatly. The links to the spacecraft generally have low data rates, on the order of 2 kbps. For deep-space spacecraft, the data rates may be as low as bits/second. Links between entities on the ground can reach the maximum speed possible with ground communications.

18.7. Mission-operations preparation

Once the satellite is complete and ready to ship to the launch site the mission-operations teams begin preparations. The following items need to be completed before the first rehearsal.

1. Telemetry database;
2. Command database;
3. Spacecraft alignments;
4. Spacecraft-mass properties;

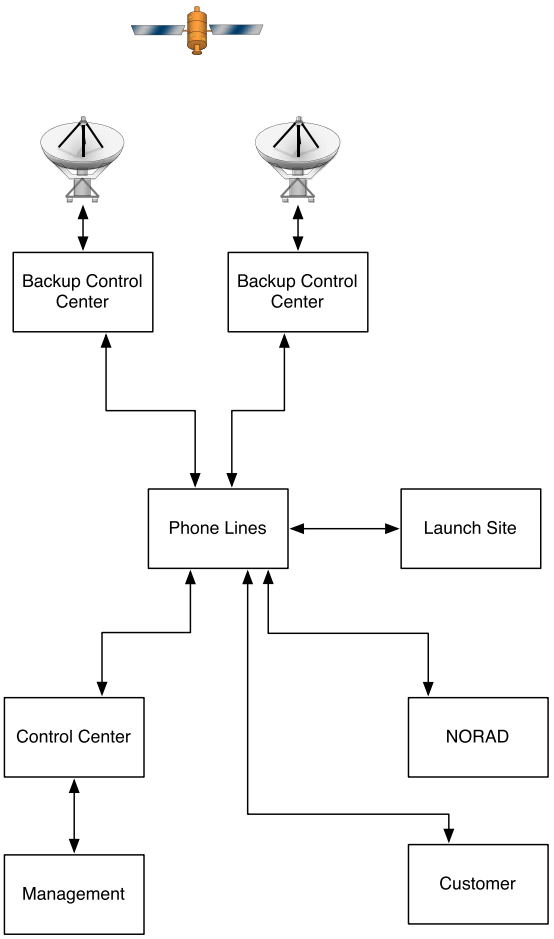


Figure 18.4 Entities involved in mission operations.

5. Spacecraft-component locations;
6. Calibrations of sensors and actuators;
7. Mission-maneuver plan;
8. Mission timeline;
9. Onorbit checkout plan;
10. Procedures handbook.

18.8. Mission-operations organization

The mission-operations organization is shown in Fig. 18.5. Spacecraft usually require 24-hour coverage during critical mission phases that include launch, transfer orbit, and

early mission-orbit operations. This is not always possible for spacecraft that do not have multiple grounds stations or intersatellite links, such as to TDRS. The launch team is organized into shifts with a mission director supervising the shift supervisors. Shifts can range from 8 to 12 hours. Under the shift supervisor are the subsystem leads and the spacecraft operator. The spacecraft operator sends commands to the spacecraft. While the software is not a subsystem there are always ground- and flight-software engineers to handle any software problems that may come up during operations. The software engineers also handle computer and networking issues both on the ground and in space. The mission-planning team performs orbit determination and determines when to fire

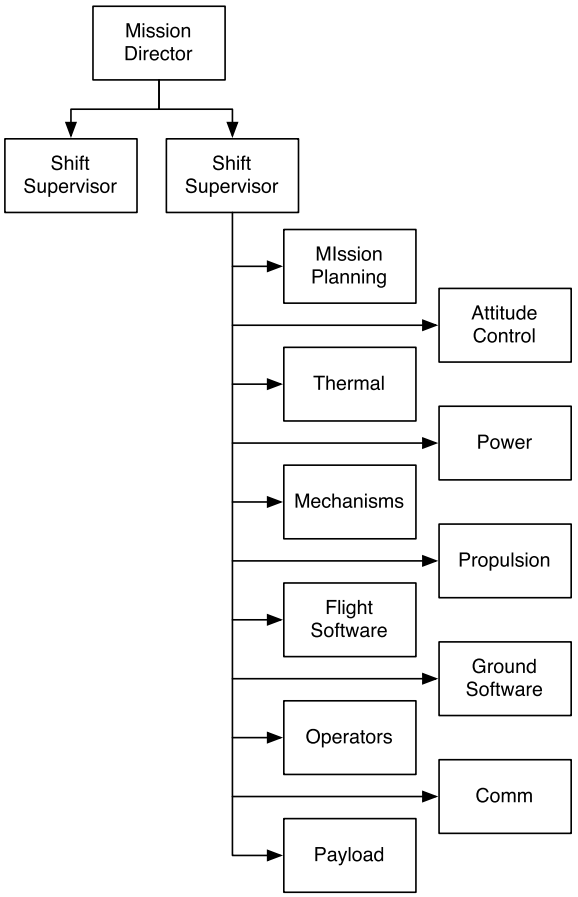


Figure 18.5 Mission-operations organization.

the thrusters to progress the spacecraft towards its final orbit. They deliver burn plans, which include thrusters to be used, burn duration, and burn start time, to the attitude control engineers who plan the maneuver. Modern spacecraft sometimes have GPS

systems or other autonomous navigation systems so they may not need ground-based orbit determination.

The attitude control engineers are responsible for the pointing of the spacecraft during all mission operations. In addition, they are responsible for the momentum state of the spacecraft and schedule momentum unloading if needed. The attitude control engineers track the fuel consumption of the attitude control thrusters, even when they are used for orbit-change maneuvers or station keeping.

The thermal engineers track spacecraft temperatures. They command heaters on or off as needed to maintain the temperatures of all components within the desired bounds.

The power engineers monitor the power state of the satellite. They command charging rates for the batteries and may command offsets of the solar wings if too much power is being absorbed.

The mechanisms engineers are primarily responsible for deployments of antennas, solar wings, etc. They usually are the engineers to command the detonation of pyroelectric bolt-cutters, etc.

The propulsion engineers are responsible for the propellant-distribution systems, that is to say, the plumbing, onboard the spacecraft. They command the firing of pyro valves that connect the fuel tanks to the thrusters. The attitude control engineers usually are responsible for the engines themselves.

The flight- and ground-software engineers correct problems inflight and ground software. The former may be fixed by uploading patches to the flight software.

The operators issue all commands to the spacecraft. The commands may be stored in lists written by subsystem engineers but the operators send each command and verify that they have been correctly received by the spacecraft.

18.9. Mission-control center

A modern mission-operations center is organized as shown in Fig. 18.6. Redundancy is not shown but the server and network would all be redundant. Generally, at least two possible ground stations are available.

18.10. Mission-operations example

A typical station-keeping maneuver is described below. It is assumed that at each step the next step is approved.

1. Mission analysts use range and range data to produce an up-to-date set of osculating orbital elements. If the spacecraft has GPS, this may be done onboard.
2. Mission analysis plans a velocity change to correct the orbit.
3. The plan is given to the shift supervisor who confers with the mission director.

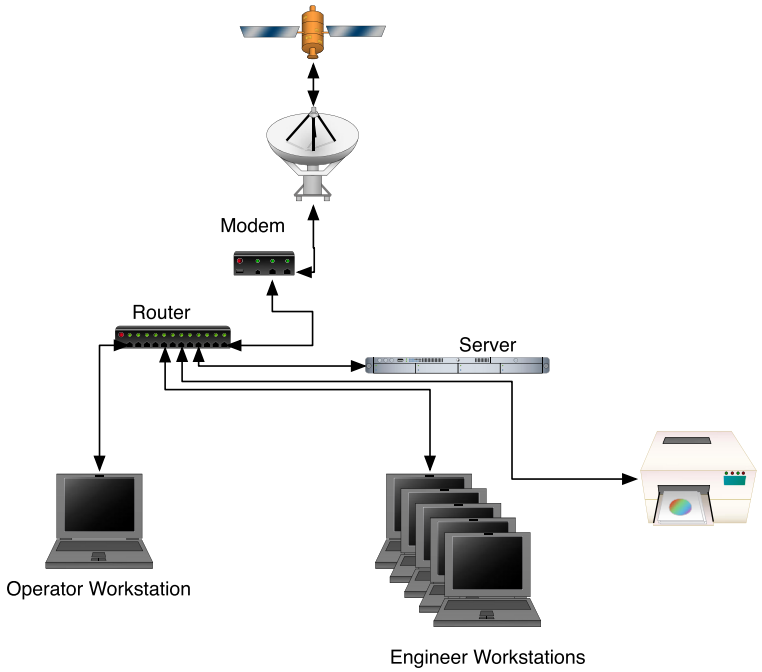


Figure 18.6 Mission operations.

4. The approved plan is given to attitude control. The attitude control analysts prepare the command lists. The spacecraft will transition into station-keeping mode, thrusters and gyros need to be enabled and gyro and thruster heaters enabled to warm up the devices.
5. Thermal and power engineers review the timing of the plan to ensure that no temperature or power limits will be exceeded.
6. Thermal and power engineers submit command lists to the operator to adjust battery rate of charge and heater status. They monitor telemetry to verify the successful completion of the commands.
7. The spacecraft operator instructs the ground station to increase transmission power.
8. The attitude control engineers submit a command list to enable thrusters and gyros. They monitor telemetry to verify the successful completion of the commands.
9. The attitude control engineers submit a command list to load station-keeping parameters. They monitor telemetry to verify the successful completion of the commands.
10. The station-keeping abort command list is preloaded into the command consoles at the command center and supporting ground stations.
11. The attitude control engineers submit a command list to start station keeping.

12. All subsystem engineers monitor their subsystems to make sure all telemetry stays within desired limits.
13. Station keeping terminates. Gyros and thrusters are disabled.
14. Attitude control engineers monitor transition into normal mode.
15. Transmission power is reduced.
16. The operator begins periodic range and rate collections.
17. Mission analysts compute the new orbit using assumed burn parameters. The actual burn parameters will be estimated as part of orbit determination.

For many organizations, commands need to be approved by one or more layers of management. This may reduce errors but will also slow down reaction time to unforeseen events. In some cases, the decision to send a command may require days of meetings. With the advent of sophisticated integrated simulations, simulations may be conducted to verify an operation sequence before operating.

CHAPTER 19

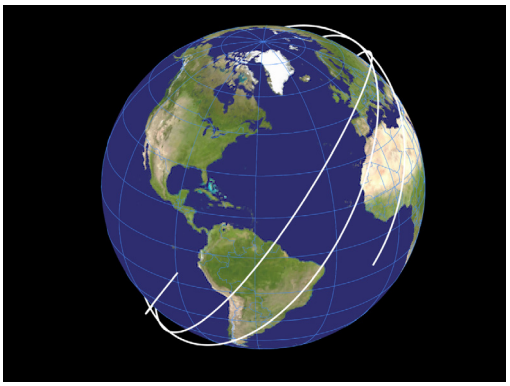
Passive control-system design

19.1. Introduction

Many satellites do not require tight pointing. In some spacecraft, passive control systems can meet the pointing requirements. This chapter discusses a design that uses a torque rod and magnetoresistive dampers.

A very important point is that magnetoresistive dampers do not work without the torque rod. The spacecraft will align the torque rod with the magnetic-field vector. The dampers then damp rates about that orientation. The torque rod only assures alignment with the magnetic-field vector. We can design the spacecraft to have a preferred orientation via the gravity gradient to get a desired Earth-pointing attitude.

```
%% Compute the ISSOrbit

[el, jD0] = ISSOrbit;
p         = Period(el(1));
t         = linspace(0,3*p,1000);
r         = RVOrbGen(el,t);

PlotOrbit(r,t,jD0)
```



Example 19.1: ISS orbit in the Earth-fixed frame.

19.2. ISS orbit

The CubeSat will separate from the International Space Station (ISS). The ISS orbit is given in Table 19.1.

The corresponding two-line elements are

```
ISS
1 25544U 98067A    20092.51597385  .00016717  00000-0  10270-3 0  9025
2 25544  51.6440   7.8556 0005107  74.2499 285.9214 15.48942018 20129
```

The two-line elements are defined in Table 19.2. The decimal point is assumed in some of the terms.

Two-line elements change with time so it is best to use the latest.

https://doi.org/10.1016/B978-0-32-399915-1.00031-0

Table 19.1 ISS orbit.

Element	Value
Semimajor axis	6799.7 (km)
Inclination	51.7 (deg)
Ascending Node	7.9 (deg)
Argument of Perigee	78.1 (deg)
Eccentricity	0.0014
Mean Anomaly	-77.9 (deg)
Epoch	2 458 941.0 (days)

Table 19.2 Two-line elements.

Line	Element	Description
1	1	Satellite number
1	2	Classification (The U for unclassified)
1	3	International Designator (Last two digits of launch year)
1	4	International Designator (Launch number of the year)
1	5	International Designator (Piece of the launch)
1	6	Epoch Year (Last two digits of year)
1	8	Epoch Day (Fractional Day of year)
1	9	First Time Derivative of Mean Motion divided by six
1	10	Second Time Derivative of Mean Motion divided by six
1	11	BStar drag term
1	12	The number 0
1	13	Element number
1	14	Checksum
2	1	Satellite number
2	2	Inclination (deg)
2	3	Right ascension of the ascending node
2	4	Eccentricity
2	5	Argument of perigee (deg)
2	6	Mean anomaly (deg)
2	7	Mean motion (revs/day)
2	8	Revolution number at epoch (revs)
2	9	Checksum

19.3. Gravity gradient

Our spacecraft will be a 1U CubeSat. The CubeSat model is shown in Fig. 19.1. This model is generated using the GGCubeSatModel.m script, that uses MATLAB® CAD modeling functions from the Spacecraft Control Toolbox. The primary purpose of this CAD model is to create the geometric configuration of the vehicle to support disturbance calculations and for computing the mass properties. It helps us to locate the

boom position to produce the desired gravity-gradient stability. The CAD tools are also helpful in laying out the spacecraft components.

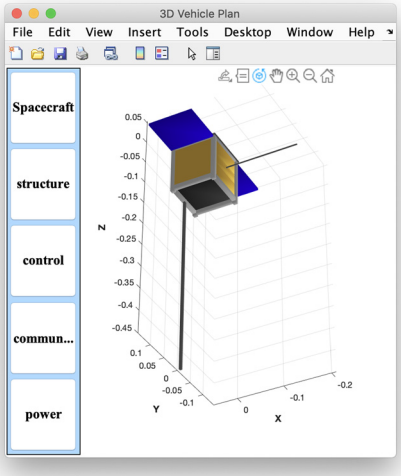


Figure 19.1 CubeSat model.

The script can also be run with the antenna, boom, and panels undeployed as shown in Fig. 19.2.

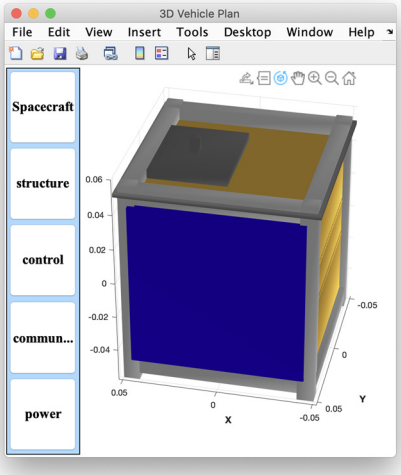
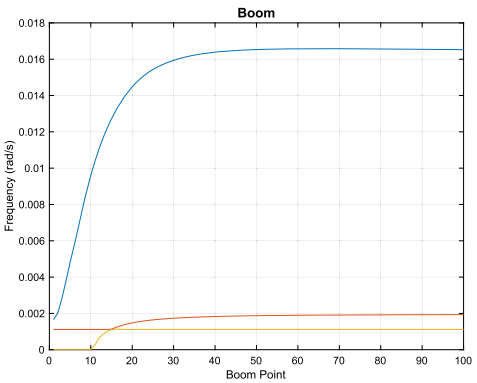


Figure 19.2 CubeSat prior to deployment. The box at the top is the folded boom with the tip mass.

The boom is offset in x to get the required libration frequencies. It also makes it easier to fold up for launch. The frequencies are shown in Example 19.2 as a function of boom length. Beyond that, there is no advantage to a longer boom. The magnetometer is also located on the boom to keep it away from the magnetic fields on the spacecraft.

```
%% Determine the libration frequencies

el  = ISSOrbit;

inr = [0.0013          0   -0.0002;...
             0    0.0013   -0.0000;...
       -0.0002   -0.0000    0.0006];

m   = 100;
r   = [0.45*ones(1,m);zeros(1,m);linspace(0,4.5,m
      )];
n   = 2*pi/Period(el(1));
GravityGradientBoom( inr, n, r, 0.2 )
```



Example 19.2: Libration frequencies.

The CubeSat attitude control components are shown in Fig. 19.3. The four magnetic-hysteresis dampers are aligned along the xy diagonals on the inside of the bottom face shown, which is the $-z$ face of the body. The torque rod is aligned along the z-axis, and the magnetometer is mounted on the inside of the $+x$ face of the body.

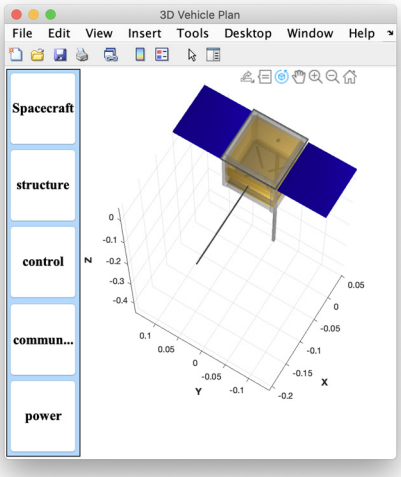


Figure 19.3 CubeSat attitude control components.

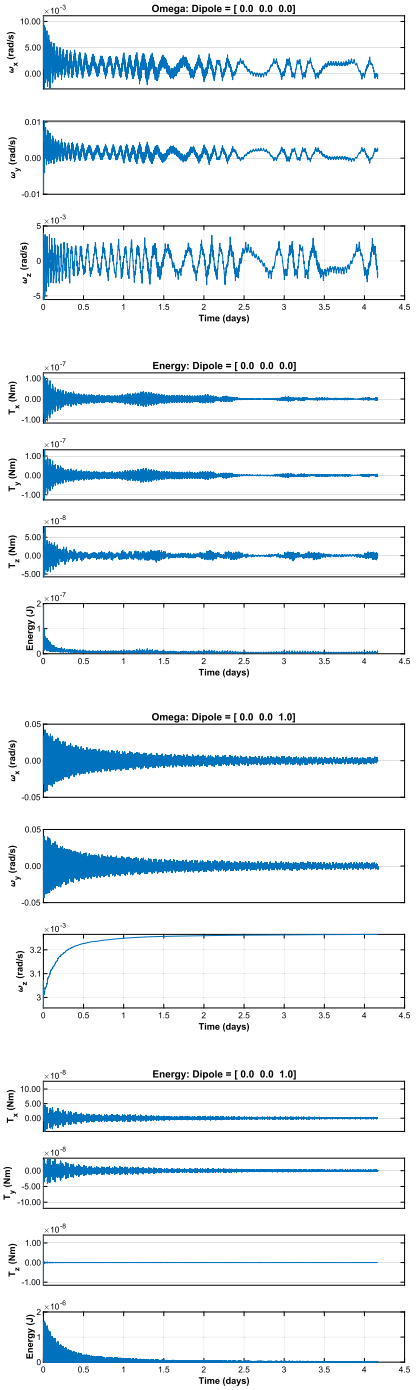
19.4. Simulations

There are two cases of interest;

1. With the torque rod;
2. Without the torque rod.

In both examples, the spacecraft will start with an angular rate. Example 19.3 shows both. Without the dipole energy, the angular rate does not converge. The angular rate converges to a steady rate with the dipole, and energy is efficiently dissipated.

```
%% Constants
secInDay      = 86400;
mu0           = 4e-7*pi;

%% User inputs
dateStart     = [2023 6 1 0 0 0];
tDuration     = 100*3600;
q0            = [1;0;0;0];
dT            = 2;
omega0        = [0.001;-0.0002;0.003];
d             = RHSRigidBodyWithDamping;
d.dipole      = [0;0;1];
[d.inertia, d.mass]      = InertiaCubeSat('1U');
cDamp         = 0.0001;

%% Start date
jDStart = Date2JD(dateStart);

% Orbit
[el,jD0] = ISSOrbit;
el(6)     = el(6) + (jDStart-jD0)*secInDay*2*pi/
        Period(el(1));
[r,v]     = El2RV(el);

% Time vector
n  = ceil(tDuration/dT);
t  = linspace(0,tDuration,n);
jD = jDStart + t/secInDay;

d.dampingType = 0;
d.dampingData.Br = 0.004;
d.dampingData.Bs = 0.025;
d.dampingData.Hc = 12;
% Damper rod unit vectors
d.dampingData.u    = [0 0 0 1 1 1;1 1 1 0 0 0;0 0
        0 0 0 0];
% Dimensions are radius 1 mm by 95 mm
d.dampingData.v = pi*0.001^2*0.095*ones(1,size(d.
        dampingData.u,2));
d.dampingFun = @RHSHysteresisDamper;
uECI = QTForm(q0,d.dampingData.u);
[bI,bIDot] = BDipole(r,jDStart,v);
hMag = Dot(uECI,bI    )/mu0;
hMagDot = Dot(uECI,bIDot)/mu0;
z = BFromHHysteresis( hMag, hMagDot, d.
        dampingData )';
x = [r;v;q0;omega0;z];
xP = zeros(20,n);

%% Simulation loop
for k = 1:n
  % Plotting
  [~,p] = RHSRigidBodyWithDamping( x, t(k), d );
  omega = x(11:13);
  energy = 0.5*omega'*d.inertia*omega;
  xP(:,k) = [x(7:end);p.torqueDipole;p.
        torqueDamper;energy];
  % Integrate
  x = RK4(@RHSRigidBodyWithDamping,x,dT,t(k),d);
end

%% Plotting
yL  = [d.states(:)' {'M_x (ATM^2)'} {'M_y (ATM^2)
        '} {'M_z (ATM^2)'} ...
{'T_x (Nm)'} {'T_y (Nm)'} {'T_z (Nm)'} {'Energy (
        J)'}];

s = sprintf('Omega: Dipole = [%4.1f %4.1f %4.1f]'
        ,d.dipole);
TimeHistory(t,xP( 5: 7,:),yL(11:13),s);
s = sprintf('Energy: Dipole = [%4.1f %4.1f %4.1f]
        ',d.dipole);
TimeHistory(t,xP(17:20,:),yL(23:26),s);
```



Example 19.3: Energy dissipation. The first two plots show the behavior without a dipole.

CHAPTER 20

Spinning-satellite control-system design

20.1. Introduction

This chapter describes how to create a preliminary control-system design for a spinning spacecraft. In this case, we will look at a satellite that is spin-stabilized in transfer orbit.

20.2. Spinning-spacecraft operation

Acquisition is the transition between transfer orbit and mission orbit. In transfer orbit, the spacecraft is spinning about its solid motor axis at the attitude needed for the delta-V burn to put the satellite in the geosynchronous orbit, or a nearby orbit from which the satellite can drift to the station. With this design, we have several options. If we add a Sun sensor we can

1. Acquire the Sun;
2. Rotate about the sunline (a safe attitude because the spacecraft is gyroscopically stable and the arrays can be easily pointed at the Sun to deliver power);
3. Acquire the Earth;
4. Lock on to the Earth;
5. Spin up the momentum-wheel assembly (MWA).

This is known as a Sun/Earth acquisition (SEA). This is desirable because the Earth is a small target at geosynchronous altitudes. Even without a Sun sensor, we can use the solar arrays as a Sun sensor of sorts. Another option is to

1. Precess the spin axis into the orbit plane;
2. Rotate in yaw until the desired orientation is achieved;
3. Spin up the MWA;
4. Lock onto the Earth.

A third option is to

1. Precess the spin-axis to orbit normal (leaving the MWA axis in the orbit plane);
2. Spin up the MWA. The spacecraft will rotate 90 degrees since the inertial momentum vector cannot be moved by internal torques;
3. Lock onto the Earth.

This last option is known as the dual-spin turn and was pioneered by RCA Astro Electronics. This last choice does not require any inertial reference after the precession to orbit normal is complete. Measuring the spin-axis attitude before the dual-spin-turn is done using horizon sensors and a single-axis Sun sensor.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00032-2

When choosing an acquisition scheme one must pay close consideration to power. During parts of the acquisition, the solar arrays may not get much Sun and if the acquisition takes too long it could leave the spacecraft in a perilous state. Since the spacecraft is a major-axis spinner, the last option does not require any additional sensing.

20.3. Transfer orbit

There are several choices for delta-V engines:

1. Solid motor (big thrust 1000s of N);
2. Liquid-fuel motor (moderate thrust 100s of N);
3. Electric thrusters (low thrust $\ll$ 1 N).

High-thrust engines usually require that the spacecraft be spun for stability during the burn. Spacecraft with moderate thrust levels can be spun or three-axis stabilized. The cheapest option for small satellites is to use a solid motor. This design will use a solid motor.

When it comes time to fire the solid motor, the satellite will need to be pointing in the right direction. For that, the operators need attitude information and thrusters to adjust the orientation. At a minimum, it is necessary to know the inertial orientation. Usually, the Sun or Earth is used as a reference. Since the spacecraft is spinning, the spacecraft spin becomes the scanning mechanism. The cheapest sensors are single-axis Sun sensors and horizon sensors.

Thrusters are needed for station keeping, which means a set of thrusters that can provide three-axis control. It is undesirable to add additional thrusters for transfer orbit, therefore it is necessary to make certain that the thrusters are usable (i.e., are not blocked by undeployed solar arrays and antennas) for controlling the spacecraft spin rate or changing the spin-axis orientation.

Since we have chosen a spinner, the final consideration is whether the spacecraft is spinning about its major or minor axis. If the former is true we are done with the preliminary design. Otherwise, we will need a control system to damp nutation and prevent the spacecraft from spinning about its major axis.

20.4. Spinning transfer orbit

20.4.1 Dynamics

A typical spinning spacecraft in transfer orbit is composed of a fairly rigid (especially compared to the deployed configuration) core spacecraft with a significant fraction of its mass in the form of liquid propellant. Fuel motion will cause energy dissipation in the spacecraft. If the spacecraft is a major-axis spinner this generally does not pose a problem since any energy dissipation will cause the spacecraft to return to its major-axis spin state. However, if the spacecraft is a minor-axis spinner its attitude will diverge and

it will attempt to spin about its major axis. Consequently, minor-axis spinners require nutation-control systems. In this example, the spacecraft will be assumed to be a major-axis spinner, and nutation control will not be discussed further.

The equations of motion for a rigid spinning spacecraft are

$$I\dot{\omega} + \omega^{\times} I\omega = T \tag{20.1}$$

where I is the spacecraft inertia, which is assumed to be constant, and ω is the spacecraft angular velocity with respect to the inertial frame measured in the body frame. If the spacecraft is spinning about the z-axis with a constant rate ω_o the transfer functions of the linearized plant are

$$\begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix} = \frac{\begin{bmatrix} s & k_x \\ -k_y & s \end{bmatrix}}{s^2 + k_x k_y} \begin{bmatrix} T_x/I_x \\ T_z/I_z \end{bmatrix} \tag{20.2}$$

$$k_x = \frac{\omega_o(I_z - I_y)}{I_x} \tag{20.3}$$

$$k_y = \frac{\omega_o(I_z - I_x)}{I_y} \tag{20.4}$$

These reduce to a pair of single integrators if ω_o, the z-axis spin rate, is zero.

The inertial angular-momentum vector is

$$H = AI\omega \tag{20.5}$$

where A transforms a vector from the body frame to the inertial frame and the energy is

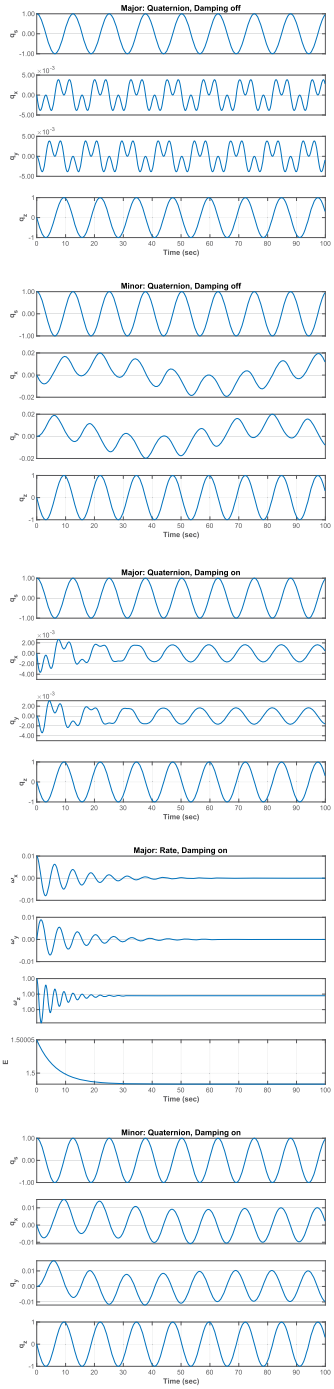
$$E = \frac{1}{2}\omega^T I\omega \tag{20.6}$$

H is fixed in magnitude and direction in the absence of external torques. Internal torques can cause E to change even though H does not. For example, when a minor-axis spinner has energy dissipation it will reorient itself so that its energy is a minimum. Even though the spin-axis changes, the angular momentum does not and it redistributes itself in the body frame so that it remains fixed in direction and magnitude in the inertial frame.

The rotation of the spin-axis about the momentum vector is often known as precession. Here, precession will mean the motion of the momentum vector with respect to the inertial frame. Since momentum is conserved, only an external torque, such as the torque produced by a thruster, can precess the momentum vector.

Example 20.1 shows the dynamics for a minor- and a major-axis spinner with and without rate damping.

```
%% Nutation damping

minor   = true;
control = true;

if( minor )
  type = 'Minor';
  inr  = diag([2 3 1]);
else
  type = 'Major';
  inr  = diag([1 2 3]);
end

dT  = 0.1;
n   = 1000;
xP  = zeros(8,n);
if( control )
  c = [0.1;0.1;0];
  damping = 'on';
else
  c = [0;0;0];
  damping = 'off';
end

x = [1;0;0;0;0.01;0;1];
for k = 1:n
  omega   = x(5:7);
  e       = 0.5*omega'*(inr*omega);
  xP(:,k) = [x;e];
  x       = RK4(@RHS,x,dT,0,inr,c);
end

t = (0:n-1)*dT;
yL = {'q_s' 'q_x' 'q_y' 'q_z'...
  '\omega_x' '\omega_y' '\omega_z' 'E' };

s = sprintf('%s: Quaternion, Damping %s',type,
      damping);
TimeHistory(t,xP(1:4,:),yL(1:4),s);
s = sprintf('%s: Rate, Damping %s',type,damping);
TimeHistory(t,xP(5:8,:),yL(5:8),s);

%% Dynamics right-hand-side
function xDot = RHS(x,~,inr,c)
q       = x(1:4);
omega   = x(5:7);
omegaDot = -inr\(c.*omega + cross(omega,inr*omega
      ));
xDot    = [QIToBDot(q,omega);omegaDot];
end
```



Example 20.1: Nutation of a minor- and major-axis spinner with and without damping.

20.4.2 Actuators and sensors

Thrusters are used to control the orientation and spin rate of the spacecraft during transfer orbit. Fixed-pulsewidth, throttleable, or pulsewidth-modulated thrusters can be used. Fixed pulsewidth is the least flexible. Both pulsewidth modulation and throttling permit the use of linear control laws, but both have threshold and saturation nonlinearities associated with them. Keep in mind that the propellant-pressurization system will affect the use of thrusters. This spacecraft employs pulsewidth modulation.

Sun sensors and horizon sensors are used for attitude determination. The Sun sensor has two outputs. Each time the Sun passes through the Sun-sensor boresight/spin-axis plane, one provides a pulse. This provides a convenient timing reference for computing spin rate. The other outputs the Sun angle in the spin-axis/boresight plane. Gyros are available, but the spin rates during transfer orbit may saturate the gyros.

The horizon sensors detect the presence of the Earth. The horizon sensors view the Earth in the CO_2 band, around 14 μm. The Earth's atmosphere is relatively stable at this frequency. When the Earth is in the sensor field-of-view the output of the sensor jumps. The horizon sensors have circuitry that detects the transition from cold space to warm Earth. There are many ways to do this. The simplest outputs a pulse whenever the output of the detector reaches a fixed threshold. Another outputs a pulse when the output reaches a fixed percentage of the peak. The latter tends to be less sensitive to Earth atmospheric-radiance variations.

Horizon sensor measurements have many error sources. These include:

1. Earth seasonal and daily radiance variations;
2. Sun and Moon interference;
3. Sensor dynamics;
4. Misalignments;
5. Irregularities in the Earth's figure;
6. and so on.

The geometry of spin-axis attitude determination is illustrated in Fig. 20.1.

20.4.3 Changing the spin rate

Two ways are available to change the spin rate. One is by manually firing the appropriate thruster(s). The second is to use the automatic spin-rate control function. This function measures spin rate using the Sun-sensor pulses and a spin-rate estimator structure and controls the spin rate with a simple proportional controller. The estimator accounts for spin-rate changes due to firing thrusters.

20.4.4 Spin-axis reorientation

Reorientations of a spinning spacecraft are performed using what is known as a spin-precession maneuver. The idea is to reorient the momentum vector without changing

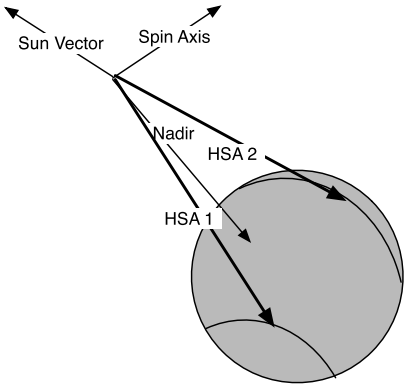


Figure 20.1 Spin-axis attitude determination diagram.

the spin rate. As only the orientation of the spin-axis is of interest, we need only concern ourselves with two angles, right ascension, and declination.

If we have continuous information about the spin-axis attitude, it is possible to perform a great-circle precession. This is illustrated in Fig. 20.2, and the spin-axis moves in the plane represented by the shaded region. α is right ascension and δ is declination. This could be performed if the satellite had three-axis gyros or if it updated its attitude from horizon and Sun-sensor measurements using a Kalman filter. A simpler approach is known as a rhumb-line precession. The geometry is illustrated in Fig. 20.3.

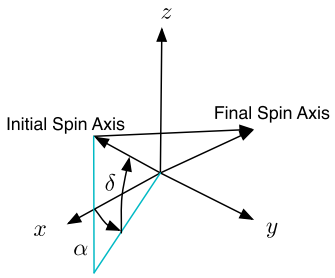


Figure 20.2 Great-circle precession diagram.

The equation for the rhumb-line angle is [1]

$$\tan\psi = \frac{\xi_s(t_f) - \xi_s(t_i)}{\log_e\left[\frac{\tan(\theta_s(t_f)/2)}{\tan(\theta_s(t_i)/2)}\right]} \tag{20.7}$$

where θ_s is the spin-axis declination and ξ_s is the spin-axis azimuth in the Sun plane, t_i is the start time, and t_f is the final time. The angle traversed is

$$\sigma = \begin{cases} |[\theta_s(t_f) - \theta_s(t_i)] \sec \psi| & \psi \neq 90^\circ \\ |[\xi_s(t_f) - \xi_s(t_i)] \sin \theta_s| & \psi = 90^\circ \end{cases} \tag{20.8}$$

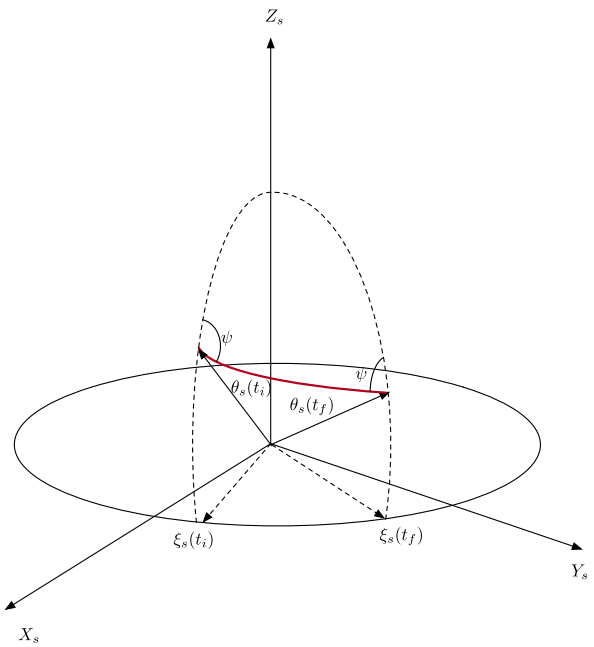


Figure 20.3 Rhumb-line precession diagram.

The rhumb-line algorithm is based on firing thrusters at a fixed time delay from the Sun pulse [1]. The thrusters are fired twice per spin period to control nutation.

The coordinate frame is such that the Sun vector is the $+Z_s$-axis. The initial and final spin-axis vectors are shown as is the path connecting them. The azimuth angle, ψ, which is the angle between the trajectory and the Sun/spin-axis plane, is a constant during the maneuver. ψ is constant because a fixed delay between the Sun crossing the spin-axis Sun-sensor plane and the thruster firing is used.

The first step is to find the initial and final unit vectors for the spacecraft spin-axis, u_i, and u_f. We then create a Sun-oriented coordinate system, where Z_s is the unit Sun vector. The other axes are found by

$$Y_s = Z_s \times u_i \tag{20.9}$$

$$X_s = Y_s \times Z_s \tag{20.10}$$

The vectors are unitized after each step. The transformation matrix from the inertial to the Sun frame is

$$C = \begin{bmatrix} X_s^T \\ Y_s^T \\ Z_s^T \end{bmatrix} \tag{20.11}$$

The transformed vectors are

$$u_{i_s} = Cu_i \tag{20.12}$$

$$u_{f_s} = Cu_f \tag{20.13}$$

The angles are

$$\theta = \cos^{-1} u_{s_z} \tag{20.14}$$

$$\xi = \tan^{-1}\left(\frac{u_{s_y}}{u_{s_x}}\right) \tag{20.15}$$

The arc tangents should use the atan2 function. The rhumb line is computed from Eq. (20.8).

Simulating a spin-precession maneuver requires a very small time step so that the timing of the torque pulses can be accurately computed. For simplicity, assume the spacecraft is spinning about its z-axis and that the sensor boresight is along the $+x$-axis. Then, the Sun pulse happens when the y component of the Sun vector in the body frame changes sign.

We want to fire a thruster to produce an angular rate about an axis perpendicular to the spin-axis and at the angle ψ. This is shown in Fig. 20.4.

The precession rate is based on the applied torque

$$\omega_p = \frac{T}{h} \tag{20.16}$$

where h is the angular momentum and T is the torque. The precession per pulse is based on the arc efficiency. If ω is the spin rate

$$\eta = \frac{\sin\delta}{\delta} \tag{20.17}$$

$$\delta = \frac{\omega\tau}{2} \tag{20.18}$$

This accounts for the torque not being applied exactly at the delay time.

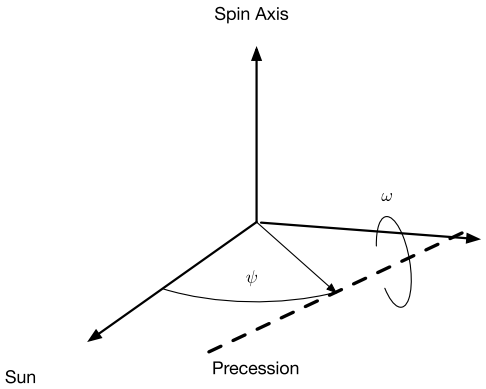


Figure 20.4 Thruster geometry for spin-precession maneuvers.

The angle change per pulse is

$$\delta\theta = \frac{\omega_p \tau}{\eta} \tag{20.19}$$

where τ is the effective pulsewidth, accounting for rise and fall times.

An example maneuver is shown in Example 20.2. The thrusters produce torques about the x-axis of the spacecraft. A pulse is fired with every Sun pulse and then one-half period later. The torque is a one-time-step approximation for the actual thruster pulse that might last for multiple time steps. In a more sophisticated simulation, the pulses would last for several time steps and have a rise and fall time.

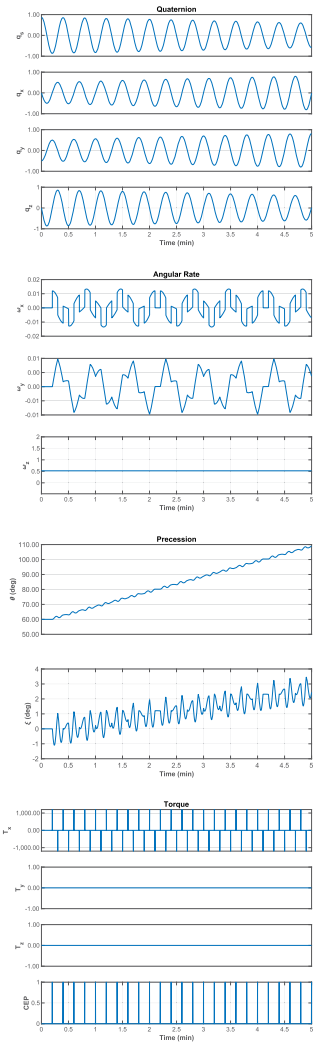
20.4.5 Attitude determination

Attitude determination is performed using three measurements, the Sun angle and the times when the horizon sensor scan crosses the edge of the Earth measured relative to the time at which the Sun crosses the Sun-sensor boresight/spin-axis plane. The three measurements are resolved into the Sun angle, the dihedral angle, and the chordwidth. The chordwidth is the difference between the trailing and leading edge times divided by the spin rate; the dihedral angle is the sum of the leading- and trailing-edge times divided by twice the spin rate. If the noise statistics for the leading- and trailing-edge times are identical, then the dihedral angle and chordwidth are uncorrelated.

Besides the measurements, attitude determination also requires the orbital position of the satellite. Consequently, accurate attitude determination requires accurate orbit determination. The attitude determination geometry is illustrated in Fig. 20.5.

The three angles give an unambiguous attitude reference. Usually, many measurements spaced over a period around apogee are taken and a least-squares algorithm is used to estimate both attitude and any biases in the data.

```
rToD    = 180/pi;
n       = 30000;
dT      = 0.01;
inr     = diag([1e3;1e3;1.3e3]);
omega   = 5*pi/30;
q       = Eul2Q([0;pi/3;0]);
x       = [q;0;0;omega];
period  = 2*pi/omega;
t       = (0:n-1)*dT;
pW      = 2*dT;
tPulse  = period/2;
zS      = Unit([0;0;1]);
uSpin   = [0;0;1];
uI      = QTForm(q,uSpin);
yS      = Unit(cross(zS,uI));
xS      = Unit(cross(yS,zS));
c       = [xS';yS';zS'];
uI      = [1;0;0];
uF      = Unit([1;0;1]);
uSunECI = zS;
[thetaI,xiI] = SpinAtt(c,uI);
[thetaF,xiF] = SpinAtt(c,uF);
num     = xiF - xiI;
den     = log(tan(0.5*thetaF)) - log(tan(0.5*
        thetaI));
azimuth = atan2(num,den);
prec    = (thetaI-thetaF)*sec(azimuth);
if ( prec < 0 )
  prec    = - prec;
  azimuth =   azimuth + pi;
end
yOld    = 0;
tPos    = inf;
tNeg    = inf;
tDelay  = azimuth/period/(2*pi);
xP      = zeros(13,n);
for k = 1:n
  [cEP,yOld] = SSA(x(1:4),uSunECI,yOld);
  torque = [0;0;0];

  % Fire every half period
  if( cEP )
    tPos     = t(k) + tDelay;
    pulsePos = true;
  end

  if( t(k) > tPos && pulsePos )
    tNeg     = t(k) + period/2;
    torque   = [1200;0;0];
    pulseNeg = true;
    pulsePos = false;
  end

  if( t(k) > tNeg && pulseNeg )
    torque   = -[1200;0;0];
    pulseNeg = false;
  end

  [theta,xi] = SpinAtt(c,QTForm(x(1:4),uSpin));
  xP(:,k)    = [x;theta;xi;torque;cEP];
  x          = RK4(@RHS,x,dT,0,inr,torque);
end

yL = {'q_s' 'q_x' 'q_y' 'q_z' '\omega_x' ...
'\omega_y' '\omega_z' '\theta_(deg)' ...
'\xi_(deg)' 'T_x' 'T_y' 'T_z' 'CEP'};

k = 1:4;
TimeHistory(t,xP(k,:),yL(k),'Quaternion');
k = 5:7;
TimeHistory(t,xP(k,:),yL(k),'Angular Rate');
k = 8:9;
TimeHistory(t,xP(k,:)*rToD,yL(k),'Precession');
k = 10:13;
TimeHistory(t,xP(k,:),yL(k),'Torque');
Figui
%% Dynamics
function xDot = RHS(x,~,inr,torque)
q        = x(1:4);
omega    = x(5:7);
omegaDot = inr\(torque - cross(omega,inr*omega));
xDot     = [QItoBDot(q,omega);omegaDot];
end
%% Angle calculation
function    [theta,xi] = SpinAtt(c,uSpin)
uS    = c*uSpin;
theta = acos(uS(3));
xi    = atan2(uS(2),uS(1));
end
%% Sun sensor
function [cEP,y] = SSA(qECIToBody,uSunECI,yOld)
uSunBody = QForm(qECIToBody,uSunECI);
y = uSunBody(2);
if( y > 0 && yOld < 0 )
  cEP = true;
else
  cEP = false;
end
end
```



Example 20.2: Spin-precession maneuver.

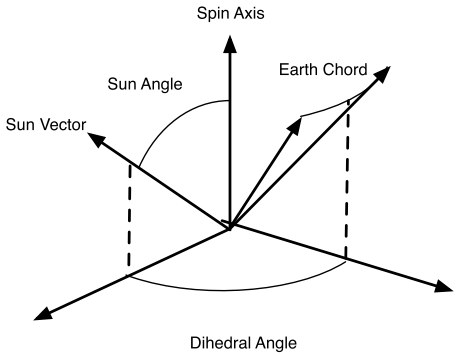


Figure 20.5 Attitude determination geometry diagram.

20.4.6 Delta-V engine firing

The delta-V engine is fired near apogee to put the spacecraft into the geosynchronous orbit. Since the spacecraft is rarely at its station at the time of the firing, the maneuver is planned to leave the spacecraft in a slightly eccentric orbit so that it will drift to its final station. The firing of the AKM causes nutation and a slight spinup of the spacecraft due to center-of-mass misalignments.

References

[1] M.H. Kaplan, Modern Spacecraft Dynamics and Control, Wiley, 1978.

CHAPTER 21

Geosynchronous-satellite control-system design

21.1. Space story

Shortly after founding Princeton Satellite Systems, Doug Freesland contacted me and asked if we could design a momentum-bias control system for their new communications satellite, Indostar-1, for Indonesia. I said we could for 750$USD. He said they only had 230$USD. Since it was a cool project I said sure and we proceeded, with the help of engineers at his company, probably the lowest-cost communications-satellite control system on record.

21.2. Introduction

This chapter describes how to create a control-system design for a geosynchronous communications satellite. It describes how to interpret customer requirements, select actuators, and sensors, and organize the design. The design that is described flew on the Indostar-1 communications satellite.

21.3. Requirements

A simplified set of requirements for a communications satellite might be:

1. The beam center shall not deviate from nominal by more than 0.2 deg.
2. The spacecraft shall stay within 0.1 deg of the station for its entire life.
3. The spacecraft's life shall be greater than 10 years.
4. No single-point failure shall cause a pointing error of greater than 0.2 deg.
5. The payload consists of a global beam pointing at nadir. This is a beam that covers a very wide area (for example, the continental United States). In addition, there is a single-spot beam.

These are a minimal set of requirements and give the designer a lot of flexibility. However, there is usually a need to meet these requirements with the lowest possible cost.

The requirements only cover mission-orbit performance. It is also necessary to get the satellite to its mission orbit. If the satellite can ride on a Proton launch vehicle, it may not be necessary to worry about transfer orbit. Future spacecraft that use electric propulsion for transfer orbit will also not have a distinct transfer-orbit mode. Otherwise, there are three distinct modes:

1. Transfer orbit;

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00033-4

2. Acquisition;
3. Mission orbit.

Transfer-orbit and acquisition requirements, besides getting the satellite safely to its mission orbit, are to minimize the fuel consumption since fuel is usually the life-limiting factor in a satellite design. Mission-orbit mode is often divided into station-keeping mode and normal mode.

21.4. The design process

The following design tasks can generally be conducted in parallel:

1. Simulation design;
2. Transfer-orbit control design;
3. Mission-orbit control-system design.

It is important to start the simulation design right away since simulations take a long time to test and debug. One generally should start by building the simplest possible rigid-body simulation with ideal sensors and actuators. The fidelity of the simulation can always be increased later as the design progresses.

21.5. Mission-orbit design

For this design, electric-heater-augmented hydrazine thrusters (electrothermal) will be used for inclination control, which is known as North/South station keeping. These thrusters are cost effective because inclination control requires a significant mass of propellant and the higher specific impulse of these thrusters (due to the electrical augmentation) reduces fuel consumption. These thrusters typically have thrust levels in the 0.5-N range. Attitude control thrusters will be needed to control the attitude during these maneuvers since the thrusters used for station keeping cannot be pulsed.

This leads to a decision on the mission-orbit architecture. There are three choices:

1. Spin-stabilized;
2. Momentum-bias;
3. Three-axis operation.

The first two are identical except that in the first the spinning part has larger inertia than the despun part. In the second, the opposite is true. The major advantage of a high-momentum design is that it provides passive attitude control. Although inertially fixed torques will precess the spin-axis, the momentum in the system will resist those torques. In addition, the momentum in the spacecraft makes it resistant to the consequences of thruster failures.

Another advantage is that since the momentum makes it resistant to disturbances, it may be possible to forgo a yaw sensor. It is easy to sense roll and pitch with an Earth sensor. Yaw is more difficult. It can be sensed intermittently with a Sun sensor,

or continuously with a star sensor or Sun sensor/gyro combination. Forgoing a yaw sensor can save a lot of money and weight. For control, we can use thrusters, magnetic torquers, solar pressure, or wheel pivoting. For sensing, we will use an Earth sensor only during the mission orbit.

A three-axis design would use pivoted or fixed reaction wheels or thrusters for control. Reaction wheels require some mechanism for unloading momentum from the system. This can be done with thrusters, magnetic torquers, or solar pressure. Yaw sensing is required if there is no bias momentum since a body-fixed yaw disturbance cannot be sensed in roll and without yaw sensing, this would cause the attitude to diverge. On this spacecraft, the only yaw requirement is levied by the spot beam. A yaw error will translate into

$$\Delta Az = 0.035\Delta_{yaw} \tag{21.1}$$

$$\Delta El = 0.035\Delta_{yaw} \tag{21.2}$$

Thus we can tolerate much larger yaw errors than roll or pitch errors. This is because this mission does not have a direct yaw requirement, the beam is circular and the yaw error has a gain of 0.035 when applied to azimuth or elevation. This means that we can probably do without yaw sensing if we use a spinner or momentum-bias design. This might not be the case if we had crosslinks to other satellites or tighter spot beam-pointing requirements.

At this point we have a number of trades to make. Some of the possible configurations are shown in Table 21.1.

Table 21.1 Possible configurations.

Design	Roll/Yaw Control	Pitch Control	Roll/Yaw Unloading	Pitch Unloading
Spinner	Thrusters	Spin motor	N/A	Thrusters
Momentum Bias	Thrusters	MWA Motor	N/A	Thrusters
Momentum Bias	Magnetic Torquers	MWA Motor	N/A	Thrusters
Momentum Bias	Solar Pressure	MWA Motor	N/A	Solar Pressure
Momentum Bias	Pivots	MWA Motor	Magnetic Torquers	Thrusters
Momentum Bias	Pivots	MWA Motor	Thrusters	Thrusters
3-Axis	Reaction Wheels	Reaction Wheel	Thrusters	Thrusters
3-Axis	Reaction Wheels	Reaction Wheel	Magnetic Torquers	Thrusters

A careful cost trade-off should be made before selecting the configuration. For the sake of this example, assume that this has been done and the momentum bias with thruster roll/yaw control has been chosen. No yaw sensor will be used. The roll and pitch axes will be sensed using an Earth sensor. Earth sensors can be either static, consisting of a ring of thermopiles in the focal plane of the sensor, or scanning, which has an oscillating mirror that sweeps the image of the Earth across pyroelectric detectors or bolometers. The choice is one largely of cost.

A station-keeping control system will also be needed. Thrusters will be used as actuators but the choice of sensors is still open. Since the disturbances will be larger (tenths of μN versus hundreds of μN) a faster-acting control system will be needed. Earth sensors tend to be noisy, thus it is difficult to differentiate the signal to get a rate measurement for a high-bandwidth control system. As a consequence, it is a good idea to use gyros to produce rate information for station keeping. Since the gyros will only be used during station keeping, they need not be long-life gyros. While the Earth sensor can still be used for angle information in pitch and roll, the yaw gyro output must be integrated for yaw. This cannot give an absolute yaw measurement, only one relative to the original yaw estimate when the gyro was initialized.

21.6. The geometry

The orbital geometry of a spacecraft with its solar arrays deployed is shown in Fig. 21.1. The solar arrays are deployed from the North and South faces of the spacecraft (which is positive and negative orbit normal). When stowed, the array panels will lie flat on these surfaces. The East and West faces require space for thrusters to perform East/West station keeping. Consequently, the apogee kick motor (AKM) has to be on the anti-Earth face. The spin-axis will run through this face and point out the front of the nozzle. The Earth sensors will be on the nadir face. The antenna may get in the way of the sensors and cutouts in the dish (which should not pose much of an antenna problem if they are much smaller than the wavelength) may be necessary. During the AKM burn, the spin-axis will point at an angle from the Earth's equatorial plane roughly equal to the negative of the transfer-orbit inclination. After the AKM burn, it will be necessary to get this axis pointing at the Earth and rotating about the pitch-axis at orbit rate.

21.7. Control-system summary

The control system is summarized in Table 21.2.

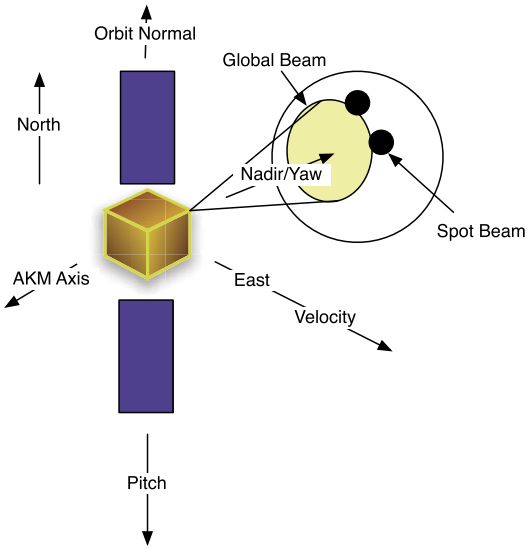


Figure 21.1 Orbital geometry.

Table 21.2 Control-system summary.

Mode	Actuator	Sensors
Transfer Orbit	Thrusters	Sun sensor and Horizon sensors
Acquisition	MWA motor	Earth sensor (for pitch)
Mission-Orbit Normal	MWA motor and Thrusters	Earth sensor
Mission-Orbit Station keeping	Thrusters	Gyros and Earth sensor

21.8. A mission architecture

The requirements: form a geosynchronous spacecraft with a mission life of 10 years. It will have a global beam that must point to nadir to within 0.2° and a one-spot beam. The goal is to design the least-expensive spacecraft that can achieve these requirements and make some money doing it! There are three phases to this mission:

1. Transfer orbit;
2. Acquisition;
3. Mission orbit.

During transfer orbit, the spacecraft is spinning about its major axis. This makes it passively stable. Reorientations are done using thrusters that precess the spin-axis. Transfer-orbit control is covered in Chapter 20.

During the mission orbit, the satellite is stabilized using a momentum wheel. The momentum in the wheel is sufficiently high so that the spacecraft is dual-spin stable. Control may be accomplished using either magnetic torquers and the momentum-wheel motor, or with thrusters. Station-keeping maneuvers are performed using thrusters. The transition between the spinning and momentum-bias states is done with the dual-spin turn. The transfer orbit, deployment, and mission orbit configurations are shown in Fig. 21.2.

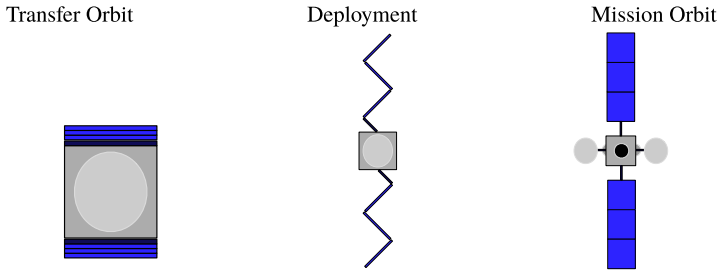


Figure 21.2 Spacecraft-configuration diagrams.

21.9. Design steps

Although a controls engineer will be concerned about such things as control-system stability margins, transient response, etc. the customer is only concerned about whether the satellite meets the requirements for beam pointing and spacecraft life. Spacecraft life is determined by the life of the components and by the amount of fuel the spacecraft carries. Component-life issues are addressed by carrying redundant components as necessary. The pointing budget and propellant budgets must be computed and updated regularly as the spacecraft design evolves.

21.10. Spacecraft overview

The spacecraft is illustrated in Fig. 21.3. Some of the features are listed below.

The spacecraft has radiators on the North and South faces. The solar wings have composite backings. This causes the wings to bend either towards or away from the Sun. In this case, the curve is approximated by a five-degree cant of the top two meters of the wings.

The geometric and surface properties are given in Table 21.3. The residual dipole is body fixed in an arbitrary direction.

The spacecraft layout was designed so that the spacecraft would be a major-axis spinner during transfer orbit and be nearly symmetrical about the Z-axis. In other words, the X-axis inertia and Y-axis inertia is nearly the same during transfer orbit.

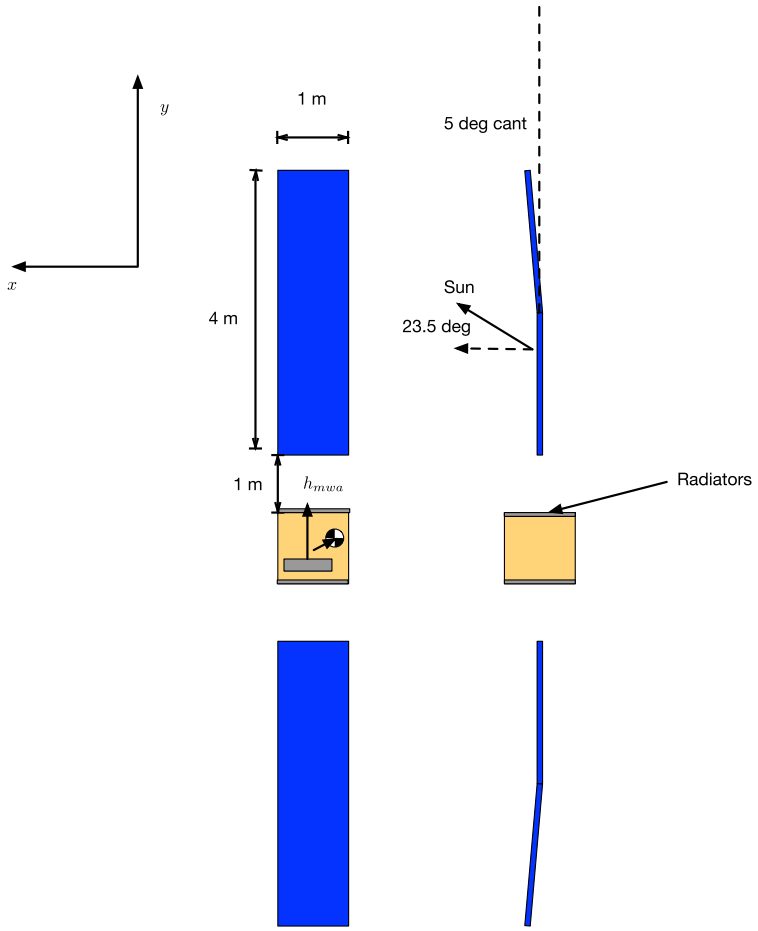


Figure 21.3 Close-up of spacecraft showing the Sun geometry.

This was accomplished by splitting the electronics into four boxes and locating them as illustrated above. Laying out a spacecraft so that it has good thermal properties, is easy to assemble, and is balanced is very difficult and expensive. A three-axis satellite stabilized in transfer orbit and did not spin at a high rate would not have to be so carefully balanced.

The antenna beams point at nadir, nearly $+Z$ in this picture. Their angle is dependent on the location of the feeds that are on the nadir panel. The fuel supply is split into two half-systems for redundancy. Two tanks are located on the East and West faces.

The thrusters are located at the corners of the box on the East, West, and North faces. The electrothermal hydrazine thrusters are clustered around the Y-axis on the

Table 21.3 Spacecraft properties.

Parameter	Value
Total Power	2077.8 W
Radiator Power	415.6 W
Solar-Array Area	8.0 m^2
Radiator Area	2.0 m^2
Residual Dipole	[2.9 2.9 2.9] ATM^2
Foil Optical [absorbed;specular;diffuse]	[0.00 0.29 0.71]
Radiator Optical [absorbed;specular;diffuse]	[0.00 0.69 0.31]
Cell Optical [absorbed;specular;diffuse]	[0.75 0.17 0.08]
Center-of-Mass	[0.03 0.02 -0.01] m
Radiator Torque	[0.01 -0.01 0.00] micro-N m

North face. Their exact location and cant angles are set to minimize losses due to plume drag. The plumbing is designed so that if the half-system fails the other half-system can perform all required maneuvers. A fixed-momentum wheel was chosen for simplicity. Pivoted wheels (either single or double) are more flexible but heavier and more expensive. They can provide higher control authority for nutation damping than the magnetic torquers.

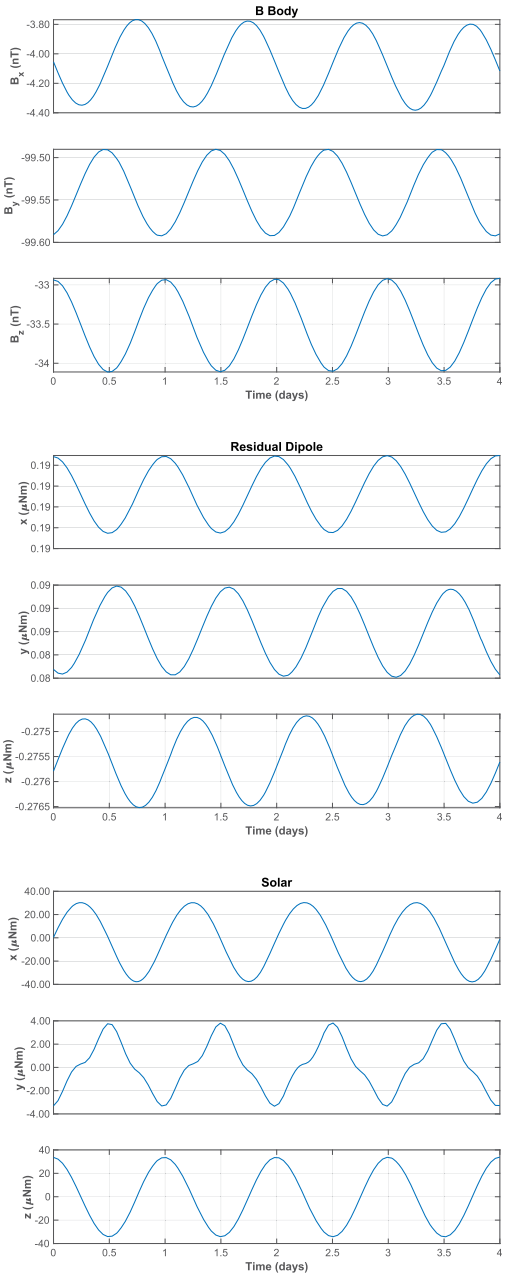
The propulsion system is a blowdown system. The tanks are filled with fuel and pressurized with helium. The mass and volume of helium determine the blowdown ratio, that is the ratio of initial pressure to empty pressure. For this spacecraft, we choose an initial pressure of 2 413 250 N/m^2 (350 psia) and a final pressure of 689 500 N/m^2 (100 psia).

The thruster layout is illustrated in Fig. 21.4. Thrusters 1 through 12 are monopropellant hydrazine thrusters with 5 N maximum thrust at a tank pressure of 2 413 250 N/m^2. Thrusters 12 through 16 are electrothermal hydrazine thrusters with a maximum thrust of 2 N at a tank pressure of 2 413 250 N/m^2.

21.11. Disturbances

The major disturbance sources on the spacecraft are solar pressure, residual dipole, and thermal torques. The model shown in Fig. 21.3 is used. Example 14.2 computes the disturbances for four orbits during the summer solstice. The Sun is 23.5 degrees out of the orbit plane. This causes a disturbance due to the canted solar wing panels. The solar panels always face the Sun, producing a xz harmonic disturbance at orbit rate. The rotation of the Sun around the bus causes the higher harmonics in y.

```
%% Setup
SetupDisturbances;

%% Disturbance model
t       = linspace(0,4*24*3600,500);
[r,v]   = RVOrbGen([42167 0 0 0 0 0],t);
jD      = jD0 + t/86400;

bECI    = BDipole(r,jD);
q       = QLVLH(r,v);
bBody   = QForm(q,bECI);
uSun    = SunV1(jD);

TimeHistory(t,bBody*1e9,{'B_x (nT)' 'B_y (nT)' 'B_z (nT)'}, 'B Body');

n = length(t);
torqName = {'Residual Dipole' 'Solar'};
torq = zeros(3,n,3);

for k = 1:n
   torq(:,k,1) = Cross(resDipole,bBody(:,k));
   torq(:,k,2) = SolarTorque(uSun(:,k),q(:,k),dSol);
end

%% Output
OutputDisturbances;

%% Solar torque
function torque = SolarTorque(uSun,q,d)
p = 1367/3e8;
tW = zeros(3,1);
for k = 1:4
   angle = acos(Unit(uSun([1 3]))'*Unit(d.uWing([1 3],k)));
   s     = sin(angle);
   c     = cos(angle);
   u     = [c 0 -s;0 1 0;s 0 c]*d.uWing(:,k);
   f     = SolarF( p, d.sCell, u, uSun, d.areaSP );
   fB    = QForm(q,f);
   tW    = tW + cross(d.rWing(:,k)-d.c,fB);
end

tB = zeros(3,1);
for k = 1:6
   uSB   = QForm(q,uSun);
   if( Dot(d.uBus(:,k), uSB) > 0 )
      f     = SolarF( p, d.sBus(:,k), d.uBus(:,k), uSB, d.areaBus );
      f     = QForm(q,f);
      tB    = tB + cross(d.rBus(:,k)-d.c,f);
   end
end

torque = tB + tW;

end
```



Example 21.1: Disturbances on a communications satellite during summer solstice. The Sun is 23.5 degrees out of the orbit plane.

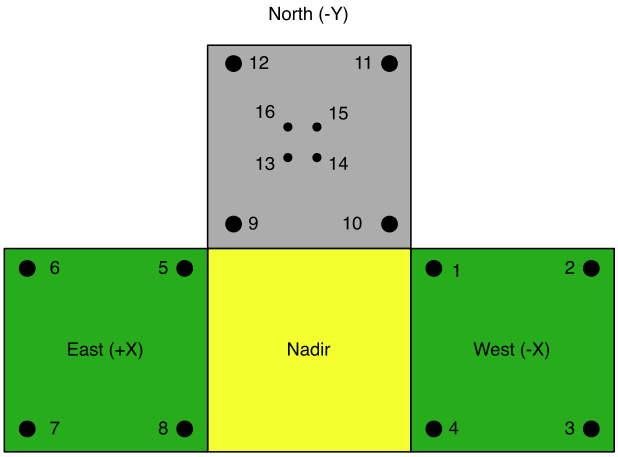


Figure 21.4 Thruster layout diagram.

21.12. Acquisition using the dual-spin turn

21.12.1 Dynamics

The dual-spin turn is based on the conservation of angular momentum. Inertial angular momentum is conserved, therefore the momentum vector will remain fixed in the inertial frame unless an external torque is applied. The spacecraft is first precessed so that the $+z$-axis (the spin-axis) is aligned with positive orbit normal. The momentum wheel that is normal to the $+z$-axis is turned on. As it spins up momentum is transferred in the body frame to the $-y$-axis. Since the angular-momentum vector is inertially fixed, the $-y$-axis aligns itself with a positive orbit normal with some residual spin rate about the $-y$-axis. For the dual-spin turn to work, the initial spin rate about the $+z$-axis must be within the limits specified by the equations

$$\frac{hW}{I_{Yaw}} \leq \Omega \leq h_W \frac{1-f}{fI_{Yaw}} \tag{21.3}$$

$$f = \left| 1 - \frac{I_{Pitch}}{\max(I_{Yaw}, I_{Roll})} \right| \tag{21.4}$$

where I_{Pitch} is the inertia of the axis with which the wheel is aligned, I_{Yaw} is the inertia of the axis about which the spacecraft is initially spinning, and I_{Roll} is the other axis inertia [1].

21.12.2 Actuators and sensors

The momentum-wheel motor is the only actuator used during the dual-spin turn. No sensing is required during the dual-spin turn. Fuel slosh provides sufficient nutation

damping to eliminate the need for a nutation damper or thruster firings to damp nutation.

21.12.3 Initialization

The dual-spin turn is initialized by despinning the spacecraft to the desired spin rate and then commanding the momentum wheel to the desired postdual-spin turn rate. The spin rate is chosen so that the total momentum of the spinning spacecraft can be absorbed by the momentum wheel.

21.12.4 Simulation

A simulated dual-spin turn is shown by typing DST([0;0;.1], 500, 'Comsat', 2). The rate about the z-axis goes to zero and the momentum is absorbed in the momentum wheel and the y-axis nutation damps to zero by the end of the simulation. In the second plot, the upper line is the commanded momentum. Fuel slosh is modeled using a damper wheel in this simulation, see Fig. 21.5.

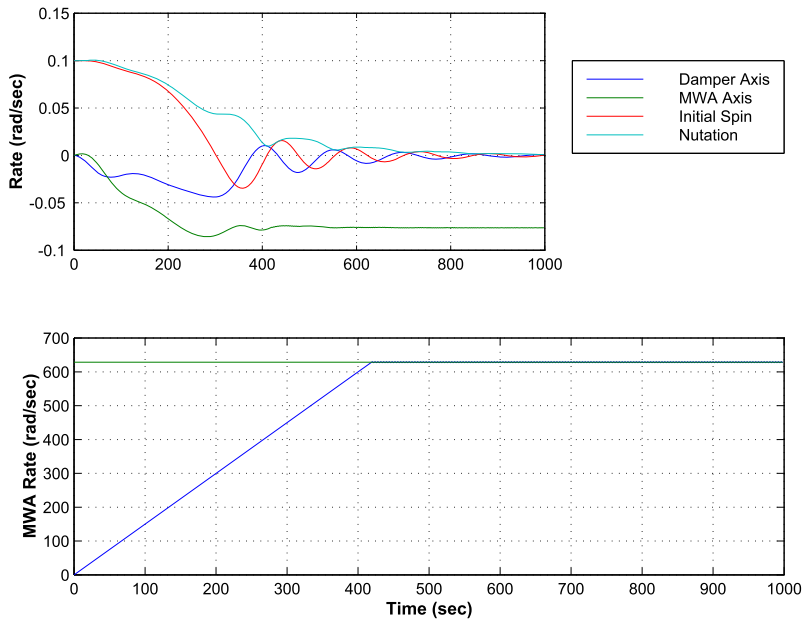


Figure 21.5 Dual-spin turn.

21.12.5 Pitch acquisition

Once the spacecraft is spinning about its y-axis and nutation has damped, the pitch-control system is activated and thrusters are selected. The Earth-sensor processing will

automatically estimate pitch rate and pitch even though the Earth is only passing through the sensor field-of-view.

21.13. Dynamics

21.13.1 Introduction

During the mission orbit, a momentum-bias spacecraft is modeled as a rigid core with a rotor and flexible solar arrays. During normal operation, the solar-array flexibility does not impact the control-system performance. Consequently, it is convenient to discuss the dynamics of the two operational modes separately.

21.13.2 Normal operations

The roll/yaw and pitch dynamics can be decoupled for analysis of normal operations. The pitch dynamics are a double integrator. The roll dynamics are a coupled fourth-order plant. There are two purely imaginary pole pairs. The first at orbit rate is due to the kinematics, the second is the nutation mode, and is due to the bias momentum. The main diagonal channels each have a purely imaginary zero pair. This zero pair is located between the orbit rate and nutation mode providing 180 degrees of phase shift and making the control problem easier.

Disturbances at orbit rate, or the nutation frequency, will cause uncontrolled growth in the attitude errors. Since many of the disturbance sources are at orbit rate, it is necessary to add an automatic control system. The nutation pole causes large oscillatory responses whenever a sudden torque is applied to the spacecraft.

The high- and low-frequency limits are of interest. At high frequencies as s approaches infinity, the system becomes a double integration. At low frequencies, the orbit-rate pole pair dominates. The attitude kinematics come from the small-angle approximation

$$\omega = \dot{\theta} + (1 - \theta^{\times})\nu \tag{21.5}$$

where ν is the orbit rate vector. The dynamical equations are

$$I\dot{\omega} + \omega^{\times}(I\omega + h_W) + \dot{h}_W = T \tag{21.6}$$

For the purpose of analyzing the roll/yaw dynamics assume that h_W is constant. The orbit-rate vector is $[0 \quad -\omega_o \quad 0]^T$. Assume also that the inertia matrix is diagonal. Linearizing the dynamics, and coupling with the kinematics gives the transfer-function

matrix

$$\begin{bmatrix} \theta_x(s) \\ \theta_y(s) \end{bmatrix} = \frac{\begin{bmatrix} I_z s^2 + \omega_o h_z & (\omega_o I_x - h_x)s \\ -(\omega_o I_z - h_z)s & I_x s^2 + \omega_o h_x \end{bmatrix}}{I_x I_z (s^2 + \omega_o^2)\left(s^2 + \left(\frac{h_x h_z}{I_x I_z}\right)\right)} \begin{bmatrix} T_x/I_x \\ T_x/I_x \end{bmatrix} \tag{21.7}$$

$$h_x = h_w + \omega_o(I_y - I_z)$$
$$h_z = h_w + \omega_o(I_y - I_x)$$

The roll/yaw equations and the equation of the damper wheel are

$$\begin{aligned} I_x\dot{\omega}_x + \omega_y\omega_z(I_z - I_y) - \omega_z h_w + J(\dot{\Omega} + \dot{\omega}_x) &= T_x \\ I_z\dot{\omega}_z + \omega_y\omega_x(I_y - I_x) - \omega_y(\Omega + \omega) + \omega_x h_w &= T_z \\ J(\dot{\Omega} + \dot{\omega}_x) &= -D\Omega \end{aligned} \tag{21.8}$$

where J is the inertia of the damper wheel and D is the damping coefficient. If we lump J with I_x we can simplify the above to

$$\begin{aligned} \dot{\omega}_x + k_x\omega_z + \epsilon\dot{\Omega} &= 0 \\ \dot{\omega}_z + k_z\omega_x - \gamma\dot{\Omega} &= 0 \\ \dot{\Omega} + \dot{\omega}_x &= \sigma\Omega \end{aligned} \tag{21.9}$$

where

$$k_x = \frac{\omega_y(I_z - I_y) - h_w}{I_x} \tag{21.10}$$

$$k_z = \frac{\omega_y(I_y - I_x) + h_w}{I_z} \tag{21.11}$$

$$\gamma = \frac{\omega_y J}{I_z} \tag{21.12}$$

$$\sigma = \frac{D}{J} \tag{21.13}$$

$$\epsilon = \frac{J}{I_x} \tag{21.14}$$

The characteristic equation can be written in Evans form

$$\frac{s^2 - \left(\frac{k_x(k_z+\gamma)}{1-\epsilon}\right)}{s^2 - k_x k_z} = -\frac{\sigma}{1-\epsilon} \tag{21.15}$$

When $\sigma = 0$ the numerator polynomial gives the poles of the system. When $\sigma = \infty$ the denominator gives the poles of the system. Hence, the open-loop zeros are given

by the denominator and the open-loop poles by the numerator. The $s = 0$ pole is the damper-wheel pole and the quadratic terms give the roll/yaw poles. For stability, the open-loop zeros must be between the closed-loop poles. These equations are only valid for $J \neq 0$.

$$\begin{bmatrix} \theta_x(s) \\ \theta_y(s) \end{bmatrix} = \frac{\begin{bmatrix} 1/I_x & 0 \\ 0 & 1/I_z \end{bmatrix}}{s^2} \begin{bmatrix} T_x \\ T_z \end{bmatrix} \tag{21.16}$$

As is evident from the transfer-function matrix, high-bandwidth controllers can ignore the orbit rate and nutational dynamics. At low frequencies the plant becomes

$$\begin{bmatrix} \theta_x(s) \\ \theta_y(s) \end{bmatrix} = \frac{\begin{bmatrix} \omega_o & (\omega_o I_x/h_x - 1)s \\ (1 - \omega_o I_z/h_z)s & \omega_o \end{bmatrix}}{s^2 + \omega_o^2} \begin{bmatrix} T_x/h_x \\ T_z/h_z \end{bmatrix} \tag{21.17}$$

The diagonal terms are small and the dynamics are dominated by crosscoupling torques. Controllers concerned with attenuating low frequency (near orbit rate) can ignore the nutation mode.

21.13.3 Dual-spin stability

A dual-spin spacecraft should be passively stable as long as the momentum wheel is at its nominal speed. This can be demonstrated by assuming that energy dissipation is provided by a damper wheel.

21.13.4 Station-keeping operations

It is necessary to include the dynamics of flexible appendages, mainly the solar arrays, for station-keeping operations. There are two reasons for this. The first is that station-keeping control systems have to respond to large disturbances caused by the delta-V thrusters. Since it is desirable to minimize the average pointing error during the burn the station-keeping system should respond quickly to these disturbances. This leads to a high-bandwidth system that may interact with flexible modes of the spacecraft. The second reason is that all thrusters are nonlinear. Even throttleable thrusters have limited throttle range and a significant minimum impulse bit. The nonlinearities give rise to control spillover beyond the control-system bandwidth. Consequently, it is necessary to consider higher-frequency modes.

The simplest way to add flex modes is to assume that the spacecraft's center-of-mass does not move when the spacecraft bends. In addition, assume that the flexible part of the spacecraft is not rotating relative to the core of the spacecraft. Finally, assume that the displacements due to bending are small and that nonlinear terms, which include the bending displacement and rate, are insignificant.

The geometry of the point mass with respect to the center-of-mass is illustrated in Fig. 21.6.

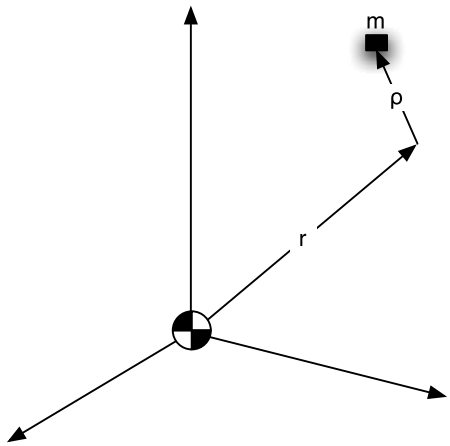


Figure 21.6 Station-keeping geometry.

Without any loss of generality, the flexible-body model consists of point masses. r is the vector from the center-of-mass to the point mass is not necessarily small. ρ is the displacement of the mass from its nominal position.

We will assume that the inertia matrix of the spacecraft includes the contributions due to the mass m and the moment arm r. Thus I is the inertia of the undeformed spacecraft. The equation for the core spacecraft angular acceleration is

$$I\dot{\omega} + \omega^{\times}(I\omega + h_w) + \dot{h}_w + \sum m_i r_i^{\times} \ddot{\rho}_i = T \tag{21.18}$$

The summation is over the physical coordinates. The acceleration of each point mass causes a torque on the spacecraft. The equation for the point-mass acceleration is

$$\ddot{\rho}_i = r_i^{\times}\dot{\omega} + \omega^{\times}\omega^{\times} r_i = \frac{f_i}{m_i} - \frac{F}{m_i} \tag{21.19}$$

These equations are quite general and can apply to any flexible appendage of any shape or size. Since the center-of-mass does not move, there is no inertial coupling between the flexible degrees-of-freedom. Coupling may be introduced through the force term on the right-hand side.

For simple systems these equations can be solved directly. For more complex systems with many flexible degrees-of-freedom it is convenient to transform the flexible degrees-of-freedom into modal coordinates. Neglect the nonlinear terms and write

$$\ddot{\rho}_i = r_i^{\times}\dot{\omega} = -k_i\rho_i - F \tag{21.20}$$

Substitute $\rho = \Phi\eta$, where Φ is the modal transformation matrix and premultiply by Φ^T

$$\Phi^T m_i \Phi \eta_i - m_i r_i^{\times} \dot{\omega} = \Phi^T k_i \Phi \eta_i - F \tag{21.21}$$

Choose Φ so that

$$\Phi^T m_i \Phi = E \tag{21.22}$$

$$\Phi^T k_i \Phi = \text{diag}(\sigma_i^2) \tag{21.23}$$

This decouples the flexible degrees-of-freedom. It also makes it easier to reduce the order of the model because it can be done based on both the frequency of the mode and its modal participation.

Example 21.2 shows a station-keeping control system with flexible modes. This only shows the roll axis.

The eigenvalues are shown in the output in the command window. Only the first four are shown.

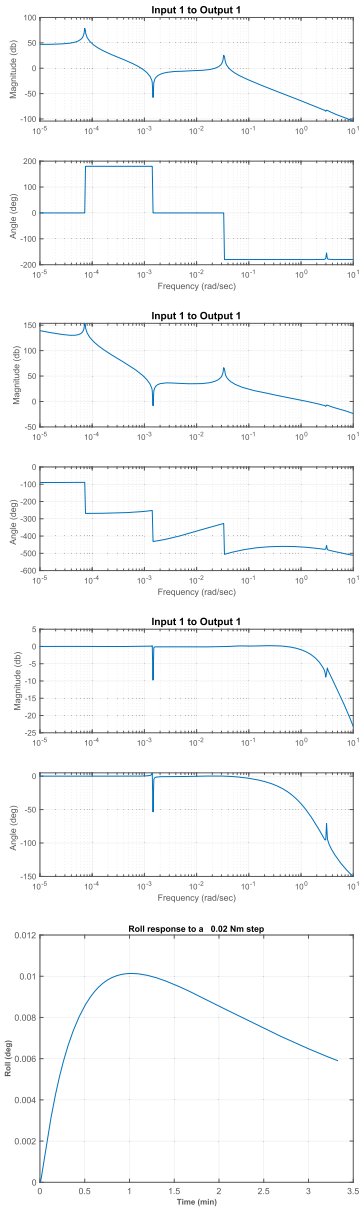
```
Open loop eigenvalues with the array at 0 deg
   0.0000e+00 + 0.0000e+00i
   0.0000e+00 + 0.0000e+00i
   0.0000e+00 + 7.2921e-05i
   0.0000e+00 - 7.2921e-05i

Closed loop eigenvalues
   0.0000e+00 + 0.0000e+00i
   0.0000e+00 + 0.0000e+00i
  -2.9060e+00 + 0.0000e+00i
  -2.4347e+00 + 0.0000e+00i
```

21.13.5 Actuators and sensors

There are two sensors used during the mission orbit. One is the Earth sensor that measures roll and pitch. The second is the rate-integrating gyro that gives integrated rates as the output. No measurement of absolute yaw attitude is available. The Earth sensor is used to measure roll and pitch during all mission-orbit modes. During station keeping, the yaw gyro is used to measure yaw. This permits a higher-bandwidth roll loop for more precise control.

Thrusters, magnetic torquers, and the momentum-wheel motor are used for control. Alternatively, a single- or double-pivoted momentum wheel could be used and the pivot motors used for control. Magnetic torquers depend on the Earth's magnetic-field vector for their control torques. The magnetic field at geosynchronous altitudes is influenced

```
%% Compensate the flexible solar array

%% Constants
degToRad = pi/180;
radToDeg = 180/pi;
dT       = 0.25;
nSim     = 800;
uStep    = 0.02;
rBModel  = false;


%% Database
inr = ComStar('MO_Inertia');

%% The state is [qX;qY;qZ;wX;wY;wZ;modes]
w = logspace(-5,1,400);

%% Load the flex model
load FlexM00

%% Open loop eigenvalues
disp('Open loop eigenvalues with the array at 0
      deg');
disp(eig(a));

d      = 0;
n      = size(b,1);
b      = b(:,1); % Tx
c      = [1 zeros(1,n-1)];

FResp(a,b,c,d,1,1,w,'unwrap');
Rename('Roll Open Loop');

%% Design the PID
zeta   = 4;
omega  = 0.3;
tauInt = 200;
omegaR = 10*omega;

[aC, bC, cC, dC] = PIDMIMO( inr(1,1), zeta, omega
      , tauInt, omegaR );

if( rBModel  )
  a = [0 1;0 0];
  b = [0; 1/inr(1,1)];
  c = [1 0];
  d = 0;
end

%% Append the plant and the controller
[a,b,c,d]         = Series(a,b,c,d,aC,bC,cC,dC);

FResp(a,b,c,d,1,1,w,'unwrap');
Rename('Roll Open Loop with Compensator');

aCL = a - b*c;
FResp(aCL,b,c,d,1,1,w,'unwrap');
Rename('Roll Closed Loop with Compensator');

%% Display the eigenvalues
s = eig(a - b*c);
DispWithTitle(s,'Closed loop eigenvalues');

%% Simulate the loop
yPlot    = zeros(1,nSim);
n        = length(aCL);
x        = zeros(n,1); % Rest conditions at the
      start
c        = [1 zeros(1,n-1)];

[aCL, b] = C2DZOH( aCL, b, dT );

for k = 1:nSim
   yPlot(k) = c*x;
   x        = aCL*x + b*uStep;
end

titleStr = sprintf('Roll response to a %6.2f Nm
      step',uStep);

t = (0:(nSim-1))*dT;
TimeHistory(t,yPlot*radToDeg,{'Roll (deg)'},
      titleStr);

Figui;
```



Example 21.2: Roll response for a station-keeping control system.

primarily by the Sun. As a consequence, it varies diurnally and with the solar cycle. During periods of intense solar activity, the field can even reverse direction.

The main advantage of magnetic torquers is that they do not change the satellite's velocity and they do not consume any propellant. Of course, their mass, power, telemetry, and command impacts must be considered. It is often better to use thrusters for momentum unloading and roll/yaw control.

The momentum-wheel motor is used to control the pitch axis. If there are constant-pitch torques, the momentum wheel will eventually saturate and the momentum-wheel motor will be unusable until momentum is unloaded with thrusters.

Thrusters are used to change the velocity of the spacecraft and are used when either the magnetic torquers or momentum-wheel motor cannot be used. This can happen during magnetic-field disturbances or when the errors are too large for the torquers or motor to control. The disadvantage of using thrusters is that they consume fuel and that the control threshold for chemical thrusters (e.g., hydrazine) is quite large, making precise control difficult.

21.13.6 Control-system organization

The flow of the control system is

1. Sensor-interface processing;
2. Error computation;
3. Control-torque computation;
4. Control-torque distribution;
5. Actuator-interface processing.

The interface processing is the low-level conversion to and from hardware data. The error computation takes the raw sensor data (now in physically meaningful form) and subtracts biases, misalignments, etc. to produce control system errors. These are processed by the control laws to produce desired torques. The control torques are converted to pulsewidths, in the case of thrusters and magnetic torquers, and voltage, in the case of the momentum-wheel motor, by the control-torque distribution algorithms. The actuator-interface processing then converts the results into values that the hardware can read.

21.13.7 Modes

The mission-orbit control system has two modes

1. Low-bandwidth control;
2. High-bandwidth control.

Either may be implemented with the momentum-wheel motor and magnetic torquers or with thrusters. The former mode uses only the Earth sensor for sensing, while the latter mode requires the yaw gyro. When thrusters are used, the momentum wheel automatically goes to a preset speed (which can be its current speed). The high-bandwidth

control is normally used for station keeping. The low-bandwidth mode is used with thrusters to transition from station keeping to torquer/momentum-wheel motor control, or when the magnetic field is perturbed.

21.13.8 Earth sensor

Earth sensors are of three types, conical scanning, scanning, and static. The latter two are the most popular on geosynchronous satellites. A major advantage of scanning sensors is that they give roll and pitch outputs directly. The static Earth sensor has sets of thermopiles arranged in a circle about the boresight. Typically each set has three. One looks at the Earth all of the time, the second straddles the Earth, and the third looks at cold space. The former and latter are used to calibrate the straddling sensor. The calibrated temperatures of the straddling sensors are used to compute roll and pitch. The scanning sensor is quite a bit simpler to incorporate into a system so it was selected for this design.

21.13.9 Gyros

A mechanical rate-integrating gyro was chosen for the gyro. This kind of gyro is inexpensive, has the required life (if only used for station keeping), and is reasonably accurate. Low drift rates are not required for this application since the gyros are only used for station-keeping maneuvers that rarely last more than 30 minutes. Each gyro has two outputs. When it senses a zero rate it generates a 44-kHz pulse train on each line. The two lines are attached to a counter, one to the upcount and one to the downcount. When a 1 least significant bit (LSB) positive rate is sensed the upcount line pulse count increases by one pulse per cycle and the downcount line pulse count decreases by one pulse per cycle. Hence, the threshold for rate detection is equal to two pulses. The up–down counter is read at 4 Hz and the difference between the current and previous values of the counter gives the change in attitude over the sampling interval. Counter overflow must be accounted for in the computation. The integrated change in angle is then passed through a noise filter and used for yaw rate and attitude. The gyro must be initialized by passing it through a low-pass filter during the gyro-initialization phase.

21.13.10 Noise filtering

All sensors have significant amounts of random noise that should be filtered before use by the control system. This filtering is independent of any filtering that is done by the control loops themselves.

The sensors are sampled at 4 Hz. To get the maximum filtering, the noise filters are designed for a 4-Hz sampling frequency. To minimize the impact on the control system no more than 10° of phase shift is acceptable at 0.1 rad/s.

21.13.11 Momentum-wheel pitch and tachometer loops

The basis for the pitch control is a tachometer loop that maintains the speed of the momentum wheel. It is desirable to keep the wheel speed within a range of the nominal speed, normally ±10% to keep the spacecraft gyroscopically stiff. A DC motor has the transfer function

$$\frac{\omega}{T} = \frac{1}{Js + \beta} \tag{21.24}$$

where β is due to the back-emf and viscous friction in the motor. A simple control scheme is to multiply the difference between the desired speed and the measured speed by a gain K and to filter the measured speed by a first-order filter. The resulting closed-loop system is

$$\Omega = \frac{(s + \omega_T)(K\Omega_C + T|C)}{Js^2 + (J\omega_T + \beta)s + \omega_T(K + \beta)} \tag{21.25}$$

where ω_T is the filter cutoff, ω_T is chosen to provide adequate damping, and K is made sufficiently large to provide good disturbance rejection and command tracking.

The input to the tachometer loop is the wheel-speed demand. The equation for the pitch loop is

$$I\ddot{\theta} + J\dot{\omega} = T \tag{21.26}$$

Since we want a pitch equation of the form

$$I\ddot{\theta} + c\dot{\theta} + k\theta = T \tag{21.27}$$

it follows that the wheel speed demand will be

$$\Omega = \frac{c\theta + k\int\theta}{J} \tag{21.28}$$

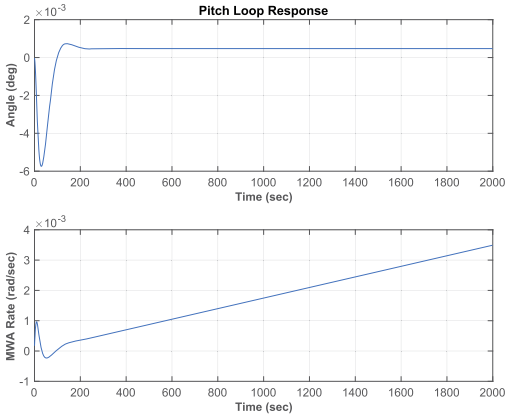
which is a proportional-integral (PI) controller.

Example 21.3 shows the pitch and tachometer loop response. The step response causes the wheel to increase in speed. Eventually, it will need to be unloaded with thrusters.

21.13.12 Low-bandwidth roll/yaw control

There are two aspects of the roll/yaw control system that must be considered. The first is that it must attenuate the external disturbances on the spacecraft. These are at harmonics of the orbit rate and generally have no significant components above twice the orbit rate. The second aspect is nutation damping. Thruster firings will tend to excite nutation. The control system must damp the nutation. Since the nutation

```
%% Tests the MWA pitch loop
inr          = ComStar('MO Inertia');
inrMWA       = ComStar('MWA Inertia');

%% Design the pitch and tach loops
zeta         = 0.7071;
wN           = 0.2;
tSamp        = 0.25;
zetaPL       = 0.7071;
wNPL         = 0.04;
beta         = 0.0;
PitchLoop(inr(2,2),inrMWA,beta,zeta,zetaPL,wN,
      wNPL,1,'Z');
```



Example 21.3: Pitch-loop response.

frequency is much higher than the orbit rate we can break the design process into two parts, one to attenuate low-frequency disturbances and the second to damp nutation.

The controller will be designed in two parts. The first part takes the low-frequency approximation to the open-loop system and selects a pair of gains to meet the pointing requirements. This approach, a purely proportional control, is the simplest. An alternative is to estimate yaw and use full-state feedback. This system is one input and two outputs but the two outputs are always in a fixed ratio to each other.

The second part is the nutation damper. We will insert a second-order compensator to stabilize nutation.

The first step is to reduce the roll/yaw equations to their low-frequency approximation. The roll/yaw equations are

$$-\frac{T_z}{h_w} = \dot{\theta}_x - \omega_o \theta_z \qquad \frac{T_x}{h_w} = \dot{\theta}_z + \omega_o \theta_x \tag{21.29}$$

Remember that

$$h = \begin{bmatrix} 0 \\ -h_w \\ 0 \end{bmatrix} \qquad \nu = \begin{bmatrix} 0 \\ -\omega_o \\ 0 \end{bmatrix} \tag{21.30}$$

The torque command is

$$\begin{bmatrix} T_x \\ T_z \end{bmatrix} = h_w \begin{bmatrix} K_{xx} \\ K_{zx} \end{bmatrix} \theta_x \tag{21.31}$$

since only roll is measured. This can be implemented with either a skew dipole or separate torquers since the ratio of the torques is fixed. Alternatively, we could have

used a dynamic compensator by having

$$\begin{bmatrix} T_x \\ T_z \end{bmatrix} = h_w \begin{bmatrix} K_{xx} & K_{xz} \\ K_{zx} & K_{zz} \end{bmatrix} \begin{bmatrix} \theta_x \\ \theta_z \end{bmatrix} \tag{21.32}$$

and estimating yaw. The latter approach requires two separate magnetic torquers.

The transfer function for the first approach is quite simple and is

$$\begin{bmatrix} \theta_x \\ \theta_z \end{bmatrix} = \frac{1}{h_w} \frac{\begin{bmatrix} s & \omega_o \\ -\omega_o + K_{xx} & s + K_{zz} \end{bmatrix}}{s^2 + K_{zx}s + \omega_o^2 - \omega K_{xx}} \begin{bmatrix} -T_z \\ T_x \end{bmatrix} \tag{21.33}$$

The disturbance torques are scaled by the bias momentum, therefore large bias momentum will reduce the attitude errors. K_{zx} provides damping. K_{xx} must be negative if it is larger in magnitude than ω_o, which will almost always be the case. The major disturbance sources will be at zero frequency and orbit rate. The transfer-function matrices are

$$\begin{bmatrix} \theta_x \\ \theta_z \end{bmatrix} = \frac{1}{h_w} \frac{\begin{bmatrix} 0 & \omega_o \\ -\omega_o + K_{xx} & K_{zz} \end{bmatrix}}{\omega_o^2 - \omega K_{xx}} \begin{bmatrix} -T_z \\ T_x \end{bmatrix} \tag{21.34}$$

The magnitude of the errors for large gains is

$$\begin{bmatrix} \theta_x \\ \theta_z \end{bmatrix} = \frac{1}{h_w} \frac{T}{\omega_o h_w \sqrt{K_{xx}^2 + K_{zx}^2}} \begin{bmatrix} 2\omega_o \\ |K_{xx}| + K_{zx} \end{bmatrix} \tag{21.35}$$

increasing either gain will decrease the roll errors. However, the two gains must be carefully balanced to attenuate yaw errors.

The estimator approach is to use full-state feedback and to estimate yaw. The estimator is

$$\begin{bmatrix} \dot{\theta}_x \\ \dot{\theta}_z \end{bmatrix} = \begin{bmatrix} 0 & \omega_o \\ \omega_o & 0 \end{bmatrix} \begin{bmatrix} \theta_x \\ \theta_z \end{bmatrix} + \begin{bmatrix} L_x \\ L_z \end{bmatrix} (z - \theta_x) \tag{21.36}$$

The transfer functions are

$$\frac{\theta_x}{z} = \frac{sL_x + \omega_o L_x}{s^2 + sL_x + \omega_o(\omega_o + L_z)} \tag{21.37}$$

$$\frac{\theta_z}{z} = \frac{sL_z - \omega_o L_x}{s^2 + sL_x + \omega_o(\omega_o + L_z)} \tag{21.38}$$

and at $s = j\omega_o$ these become

$$\frac{\theta_x}{z} = 1 \tag{21.39}$$

$$\frac{\theta_z}{z} = j \tag{21.40}$$

At orbit rate, the estimator always passes the roll measurement unfiltered to the roll estimate and phase shifts the roll measurement by 90 deg to get the yaw estimate. Note that the estimator rolls off as $1/s$, making it sensitive to unmodeled dynamics. The roll to yaw transfer function is a nonminimum phase. This limits the gain of any controller using this estimator.

Example 21.4 shows the low-frequency roll/yaw control. The last two plots show the controller performance using the disturbance model generated in this chapter.

The next step is to compensate for the nutation. First, turn the one-input–two-output system into a SISO system by appending the gain calculated above.

Several different approaches can be used to stabilize nutation. One is to use a nonminimum phase-transfer function of the form

$$\frac{s-a}{s+a} \tag{21.41}$$

The magnitude of this transfer function is always one and the phase is

$$\phi = \tan^{-1}\left(\frac{-2\omega}{\omega^2 - a^2}\right) \tag{21.42}$$

at large ω the phase is $\tan^{-1}(-0)$ or -180°. Since the magnitude of the phase shift is greater than 90°, this is known as a nonminimum phase-transfer function. Interestingly, the first-order Padé approximant to a delay of period T is

$$\frac{s-2/T}{s+2/T} \tag{21.43}$$

Thus this nonminimum phase-transfer function could be implemented as a delay.

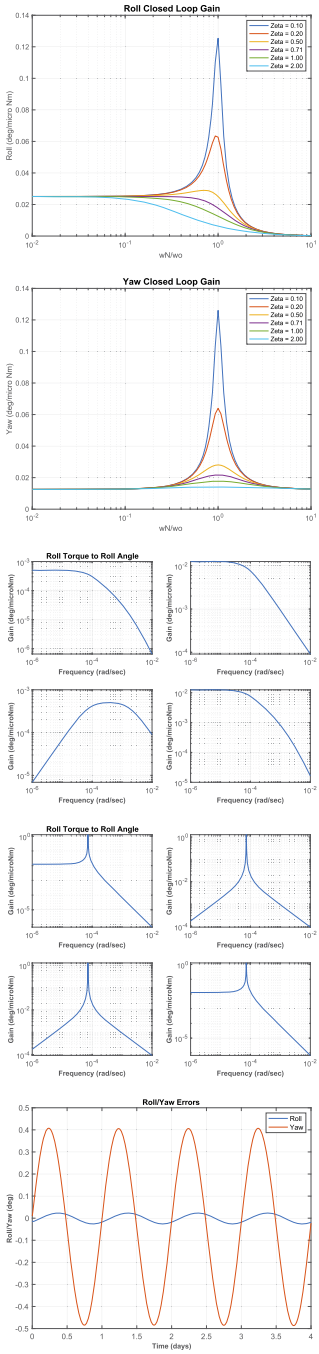
The approach presented here is to introduce a complex zero just before the nutation pole pair using the transfer function

$$\frac{s^2+\omega_z^2}{s^2+2\zeta\omega_p+\omega_p^2} \tag{21.44}$$

and a damped pole after the nutation mode. As long as the zero is at a frequency less than that of the nutation pole pair the system will be stable. Example 21.5 shows nutation damping.

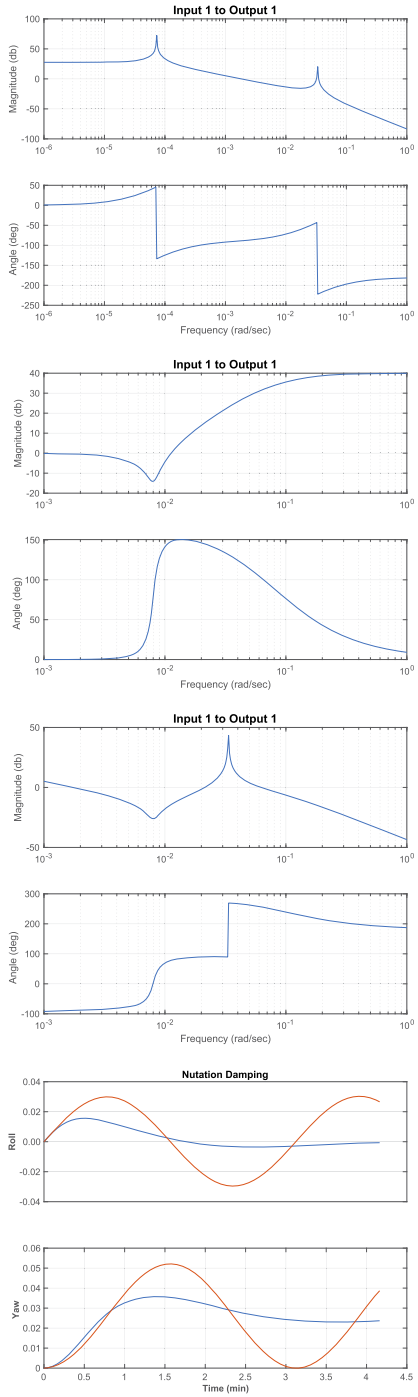
```
Eigenvalues
   -6.4909e-02  +  7.2632e-02i
   -6.4909e-02  -  7.2632e-02i
```

```
degToRad    = pi/180;
radToDeg    = 180/pi;

wo          = 2*pi/(24*3600);
hW          = ComStar('MWA_Inertia')*ComStar('
        Nominal_MWA_Rate');
iAxis       = [1 3];
inr         = ComStar('MO_Inertia');
hMWA        = ComStar('U_MWA')*hW;
wO          = ComStar('Orbit_Rate_Unit_Vector')*wo
        ;
bMin        = 75e-9;

%% Plant models
[aP,bP,cP,dP,aL,bL,cL,dL] = RYDyn( inr, hMWA, wO,
        iAxis);

n     = 100;
nZ    = 6;
zeta  = [0.1  0.2  0.5  0.7071  1.0  2.0];
k     = logspace(-2,1,n);
j     = sqrt(-1);
wN    = k*wo;
kX    = (wo^2-wN.^2)/wo;
f0    = 1.e-6/(wo*hW);
kZ    = zeros(nZ,n);
aX    = zeros(nZ,n);
aZ    = zeros(nZ,n);
leg   = cell(nZ,1);
for i = 1:nZ
   kZ(i,:) = 2*zeta(i)*wN;
   f       = f0./(j*kZ(i,:) - kX);
   aX(i,:) = abs( 2*wo.*f );
   aZ(i,:) = abs((j*wo + kZ(i,:) - j*(kX-wo)).*f);
   leg{i}  = sprintf('Zeta_=_%4.2f',zeta(i));
end


Plot2D(k,aX*180/pi,'wN/wo','Roll_(deg/micro_Nm)',
        'Roll_Closed_Loop_Gain','xlog')
legend(leg);
Plot2D(k,aZ*180/pi,'wN/wo','Yaw_(deg/micro_Nm)','
        Yaw_Closed_Loop_Gain','xlog')
legend(leg);
Plot2D(k,kZ,'wN/wo','kZ','Roll_Angle_to_Yaw_
        Torque_Gain','xlog')
legend(leg);
Plot2D(k,kX,'wN/wo','kX','Roll_Angle_to_Roll_
        Torque_Gain','xlog')

%% Skew dipole control system simulation
zeta  = 2.5;
wN    = 5*wo;
kRY   = [(wo^2-wN^2)/wo;2*zeta*wN];
aCL   = aL + bL*kRY*[1 0]*hW;
s     = eig(aCL);

dModel =  load('DisturbanceModel');

[a,b] = C2DZOH(aCL,bL,dModel.dT);

%% The torque transmission plots
titlesRY = {'Roll_Torque_to_Roll_Angle'...
            'Roll_Torque_to_Yaw_Angle' ...
            'Yaw_Torque_to_Roll_Angle' ...
            'Yaw_Torque_to_Yaw_Angle'};

wP          = logspace(-6,-2,300);
mag         = MagPlot(aCL,bL,cL,dL,1:2,1:2,wP);
TTPlots(radToDeg*1.e-6*mag,wP,titlesRY,'deg/
        microNm','Closed_Loop')
[mag,io] = MagPlot(aL,bL,cL,dL,1:2,1:2,wP);
TTPlots(radToDeg*1.e-6*mag,wP,titlesRY,'deg/
        microNm','Open_Loop')

x       = [0;0];
nSim    = size(dModel.totalTorque,2);
xPlot   = zeros(length(x),nSim);
uPlot   = zeros(2,nSim);
for k = 1:nSim
   x          = a*x + b*dModel.totalTorque([1 3],k
              );
   xPlot(:,k) = x;
   uPlot(:,k) = kRY*x(1)*hW/bMin;
end

rYGain = kRY*hW;

t = (0:(nSim-1))*dModel.dT;
TimeHistory(t,xPlot*180/pi,{'Roll/Yaw_(deg)'},'
        Roll/Yaw_Errors',{1:2})
legend('Roll','Yaw')
TimeHistory(t,uPlot,{'Dipole_(ATM^2)'},'Torquer_
        Demand',{1:2})
legend('Roll','Yaw')
Figui
```



Example 21.4: Low-frequency roll/yaw control.

```
1
2  %% Load in the data generated by TRYLF.m
3  % Which supplies a, b, c, d and rYGain.
4  load RYC
5
6  %% Connect the skew dipole and earth sensor
7  a        = aP;
8  b        = bP*rYGain;
9  c        = [1 0 0 0]; % Only roll is available
10 d        = 0;
11
12 w        = logspace(-6,0,400);
13 FResp(a,b,c,d,1,1,w);
14 Rename('Open Loop');
15
16 % Adjust these parameters to change the nutation
       damping
17 wz       = 0.008;
18 zz       = 0.1;
19 zp       = 1;
20 wp       = 0.08; %
21
22 %% The nutation compensator
23 [aN,bN,cN,dN]      = Gen2nd ( zz, wz, zp, wp );
24 w                  = logspace(-3,0,400);
25 FResp(aN,bN,cN,dN,1,1,w);
26 Rename('Nutation Compensator');
27
28 [aT,bT,cT,dT]      = Series( a, b, c, d, aN, bN,
       cN, dN );
29 FResp(aT,bT,cT,dT,1,1,w);
30 Rename('Open Loop with Compensator');
31 aCL = aT + bT*cT;
32 s = eig(aCL);
33
34 DispWithTitle(s,'Eigenvalues');
35
36 %% Simulate the nutation compensator
37 dT             = 0.25;
38 nSim           = 1000;
39 [aZCL,bZCL] = C2DZOH(aCL,bT,dT);
40 [aZOL,bZOL] = C2DZOH(a,b,dT);
41
42 xCL = [0;0;0.001;0;0;0]; % Some roll rate
43 xOL = xCL(1:4);
44
45 xCLPlot = zeros(2,nSim);
46 xOLPlot = zeros(2,nSim);
47
48 for k = 1:nSim
49    xCLPlot(:,k) = xCL(1:2);
50    xOLPlot(:,k) = xOL(1:2);
51    xCL          = aZCL*xCL;
52    xOL          = aZOL*xOL;
53 end
54
55 t = (0:(nSim-1))*dT;
56
57 TimeHistory(t,[xCLPlot;xOLPlot],{'Roll' 'Yaw'
       },...
58    'Nutation Damping',{[1 3] [2 4]})
```



Example 21.5: Nutation damping.

```
-1.4196e-02 +  2.3776e-02i
-1.4196e-02 -  2.3776e-02i
-1.7143e-03 +  0.0000e+00i
-7.6505e-05 +  0.0000e+00i
```

21.13.13 Thruster control

Thrusters are used both during station keeping and as a backup to the magnetic-torquer/momentum-wheel control system. For backup purposes, the thruster roll/yaw control uses the same compensator as the magnetic roll/yaw control. However, pitch control uses a proportional-derivative control.

21.13.14 High-bandwidth roll/yaw and pitch control

In high-bandwidth mode, each axis is controlled independently with a proportional-integral-differential controller. The yaw measurement is taken from the yaw gyro. The yaw gyro does not provide an absolute attitude reference and the control system assumes that the initial yaw attitude is zero. This does not pose a problem as long as the mode is not run for long periods.

The rigid-body equations are

$$T = I\dot{\omega} + \omega^{\times} I\omega \tag{21.45}$$

If rates are small and the angles are small this reduces to

$$T = I\ddot{\theta} \tag{21.46}$$

If we define

$$T = Iu \tag{21.47}$$

then the axes are decoupled and the control problem is reduced to

$$u = \ddot{\theta} \tag{21.48}$$

or three decoupled double integrators. These equations are valid if the flexible modes are at much higher frequencies than the control bandwidth and, in this case, the bandwidth is much higher than the nutation mode. Once the control is computed the control torques are formed by the operation

$$Iu \tag{21.49}$$

which is a 3-by-3 matrix times a 3-by-1 vector multiply. We can now design each axis control loop independently.

A PID controller is of the form

$$T = K_p + K_R \frac{\omega_R s}{s + \omega_R} + \frac{K_I}{s} \tag{21.50}$$

The derivative term is multiplied by a first-order filter to prevent differentiation at high frequencies. Rearranging gives

$$\frac{T}{\theta} = \frac{(K_P + K_R\omega_R)s^2 + (K_p\omega_R + K_I)s + (K_P + K_i\omega_R)}{s(s + \omega_R)} \tag{21.51}$$

If ω_R is set to ∞ this becomes

$$\frac{T}{\theta} = \frac{K_R s^2 + K_P s + K_I}{s} \tag{21.52}$$

This transfer function is not proper, i.e., it goes to ∞ as s goes to 0. The latter cannot be implemented in state-space form for this reason. In practice, this is not a problem if noise filtering is included in the loop.

In this system the plant is not a set of pure double integrators, instead, it has the undamped nutation pole. If the bandwidth is set high enough, the nutation pole will be unobservable by the control system. This does not mean it has disappeared and it will be excited by control activity and the disturbance torques. This would pose a problem when the PID was turned off. The normal mode thruster control will damp the nutation and so this is not a problem.

21.13.15 Magnetic-torquer control

The magnetic torquers are controlled with a timer. For each control period, the timer is passed the duration of the magnetic pulse. The pulse always starts immediately.

21.13.16 Thruster control

The thrusters are controlled with timers. For each control period, the timer is passed the duration of each thruster pulse and a delay from the issuing of the command. This permits off-pulse modulation of the thrusters and execution of spin-precession maneuvers. The control algorithms produce a three-axis torque demand. This must be translated into pulsewidth demands for the selected thrusters. The thrusters generate unidirectional torques and it is generally not possible to divide the thrusters into bidirectional pairs. The simplest approach is to use a linear programming approach to torque distribution. This is done using the simplex algorithm. Simplex solves the problem

$$T = Au \tag{21.53}$$

$$c = \sigma_{i-1}^{n} |u_i| \tag{21.54}$$

where A is a 3-by-n matrix of torque vectors, u is an n-by-1 column vector of pulsewidth fractions, c is the cost, and T is the desired torque. Simplex finds u.

This algorithm uses matrix methods to speed the computation and is optimized for three constraint equations.

21.13.17 Actuator saturation

All actuators have a saturation limit. If the compensator has integrators this can lead to sluggish behavior. The solution is to add antiwindup compensation. This is straightforward. Implement the digital controller as

$$y_k = sat(Cx_k + Du_k) \tag{21.55}$$

$$x_{k+1} = (A - LC)x_k + (B - LD)u_k + Ly_k \tag{21.56}$$

If y is not saturated this reduces to the equations

$$y_k = Cx_k + Du_k \tag{21.57}$$

$$x_{k+1} = Ax_k + Bu_k \tag{21.58}$$

If it is saturated the dynamics of the compensator are governed by the plant $A - LD$. L must be chosen so that the saturated compensator is stable. Making $A - LD$ have all of its poles at zero is a convenient choice.

21.13.18 Thruster resolution

The pulsewidth is related to the torque demand by the equation

$$\tau = \frac{u}{u_{max}} T \tag{21.59}$$

where T is the pulsing period, u is the torque demand, and u_{max} is the maximum torque. The larger the T is the larger τ will be for a given torque demand, thus increasing the pulsing period can increase the resolution. This does not mean that the control period must be increased, rather one fires a pulse every T seconds. T should be an integral multiple of the control period.

If u is much less than u_{max} this will not normally have a major impact on the phase margins for the system. It will have an impact and must be considered when compensating the loop.

An alternative is to add triangle-wave dither to the pulse demand. If the dither frequency is high enough, and the plant has low-pass filter characteristics, dither can reduce the effect of the minimum pulsewidth. Unfortunately, in a digital control system, the sampling rate puts a limit on the dither frequency; hence the dither will appear as a high-frequency oscillation. This may not pose a problem but must be accounted for

in the jitter budget. This leads to the problem of the minimum pulsewidth resolution which has a major impact on pointing performance since it leads to a large deadband around zero.

21.14. Summary

A control-system design for a momentum-bias geosynchronous communications satellite is presented. All control loops have been designed. A 13-year life and a pointing CEP of 0.17 deg meet the design specifications.

Our task was made easier by a relatively stiff array and the absence of a yaw-pointing requirement. If the array were more flexible we would have had to add compensation to the station-keeping loops to stabilize the low-frequency flex modes. If yaw were important we would have had to design a more sophisticated low-frequency yaw controller or add yaw sensing.

The next step in the design would be to add realistic actuator and sensor models and nonlinear dynamics models and verify that the design still meets requirements.

Other things to consider are:

1. Add in the actuator and sensor dynamics models.
2. Determine what commands are needed for each controller.
3. Determine how to do mode transitions (from station keeping to normal mode, for example).
4. Determine what telemetry is needed to monitor the control system.
5. Translate the MATLAB® into flight software.
6. Write operational procedures.
7. Train the spacecraft operators.

Nonetheless, this is well on the way to a complete design!

References

[1] C. Hubert, Spacecraft attitude acquisition from a spinning or tumbling state, Journal of Guidance, Control, and Dynamics 4 (2) (1981) 164–170.

CHAPTER 22

Sun-nadir pointing control

22.1. Space story

When we worked on GPS IIR at GE Astro Space, I do not think anyone had an idea of the impact it would have on society. As it was a US Air Force program, we thought mostly of military applications. The idea that millions, if not billions of people would someday be traveling with hand-held phones using GPS to find their way was not anything we imagined.

22.2. Introduction

This chapter provides the theory behind Sun-nadir pointing-control systems. A Sun-nadir pointing control system is designed to point one axis of the spacecraft at the Earth and keep the solar-array cell face normal aligned with the Sun vector. This maximizes power input while keeping the Earth-pointing payload on target. For GPS, this would be the GPS antennas. Sun-nadir pointing-control systems are generally used on spacecraft in highly inclined orbits where the changes in the Sun angle with respect to the orbit plane may be large.

A characteristic of Sun-nadir pointing spacecraft is that its yaw angle changes over the orbit. When the Sun angle is near the orbit plane the spacecraft flips nearly 180° at spacecraft noon and midnight. Since the spacecraft must rotate in pitch and yaw, this precludes the use of a momentum bias or spinning spacecraft.

This chapter deals only with Sun-nadir pointing control. Station-keeping and acquisition modes are not covered. In addition, the accompanying scripts do not include logic for reinitializing of the control loops, error handling, and other necessary functions that must be included in flight-control software.

22.3. Coordinate frames

The orbit geometry is illustrated in Fig. 22.1. The Sun angle is defined as positive above the orbit plane and its projection in the orbit plane is along the $+X$-axis. The spacecraft vector in the orbit plane is an angle α away from the Sun projection. The local vertical/local horizontal (LVLH) frame is defined with $+z$ along the nadir vector, $+y$ along the orbit antinormal, and $+x$ completing the right-hand set. The spacecraft is yawed an angle θ with respect to the LVLH frame. Finally, the solar arrays are oriented at an angle γ with respect to the body frames. When $\gamma = 0$ the $+z$ array axis (the cell face)

https://doi.org/10.1016/B978-0-32-399915-1.00034-6

is aligned with the $+z$ body axis. In this chapter, the inertial frame will be defined with the XY-axes in the orbit plane, the $+Z$-axis along the orbit normal and the $+X$-axis aligned with the Sun projection. This will not normally coincide with standard inertial frames such as the ECI or J2000.0 frames. The solar array frame is illustrated in Fig. 22.2.

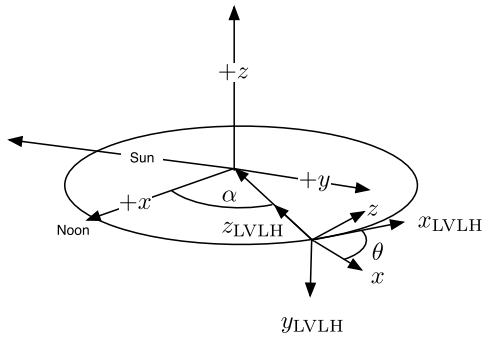


Figure 22.1 Orbit-geometry diagram.

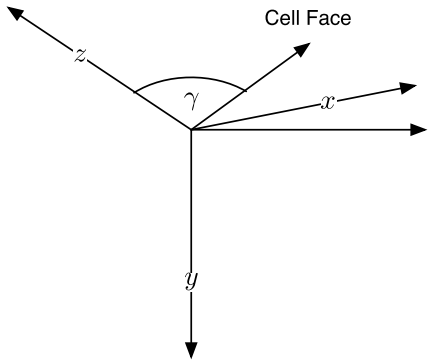


Figure 22.2 Solar-array frame.

22.4. Sun-nadir pointing

The Sun vector is represented by

$$u_s = \begin{bmatrix} -\sin\alpha\cos\beta \\ -\sin\beta \\ -\cos\alpha\cos\beta \end{bmatrix} \tag{22.1}$$

where β is the elevation angle of the Sun and α is the azimuth. The transformation matrix from the array to the core frame is

$$\begin{bmatrix} \cos\gamma & 0 & \sin\gamma \\ 0 & 1 & 0 \\ -\sin\gamma & 0 & \cos\gamma \end{bmatrix} \tag{22.2}$$

where γ is the rotation angle of the solar array. The $+z$ unit vector in the array frame is, in the core frame,

$$u = \begin{bmatrix} \sin\gamma \\ 0 \\ \cos\gamma \end{bmatrix} \tag{22.3}$$

The transformation matrix from the core frame to the LVLH frame is

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{22.4}$$

Transforming into the LVLH frame, assuming a yaw rotation of θ, and equating to the Sun vector gives

$$\begin{bmatrix} -\sin\alpha\cos\beta \\ -\sin\beta \\ -\cos\alpha\cos\beta \end{bmatrix} = \begin{bmatrix} \cos\theta\sin\gamma \\ \sin\theta\sin\gamma \\ \cos\gamma \end{bmatrix} \tag{22.5}$$

Therefore, the yaw and solar array angles are

$$\theta = \tan^{-1}(\sin\beta, \sin\alpha\cos\beta) \tag{22.6}$$

$$\gamma = \tan^{-1}(-\sqrt{(\sin\alpha\cos\beta)^2 + \sin^2\beta}, -\cos\alpha\cos\beta) \tag{22.7}$$

The two argument $\tan^{-1}$ is usually called atan2 and puts the vector in the correct quadrant in all cases. Fig. 22.3 shows the Sun-nadir trajectories.

The plots show trajectories ranging from $\beta = 0°$ to $\beta = 70°$. When $\beta = 0$, the yaw angle is always zero and the array rotates 360° every orbit. For $\beta = \pm\epsilon$, the yaw angle changes 180° at spacecraft noon and midnight. If reaction wheels are used to perform the noon–midnight turns, the peak yaw rate and yaw inertia will largely determine the momentum requirements of the system. Because of limitations in the momentum storage (or control authority if using thrusters), it will not be possible to track the ideal yaw trajectory exactly.

Assuming that the solar arrays can always be pointed so that the only error is in the yz-plane of the solar arrays, the dot product of the Sun vector in the solar-array plane and the solar-array normal will be

$$\sqrt{1 - u_y^2} = \cos\epsilon \tag{22.8}$$

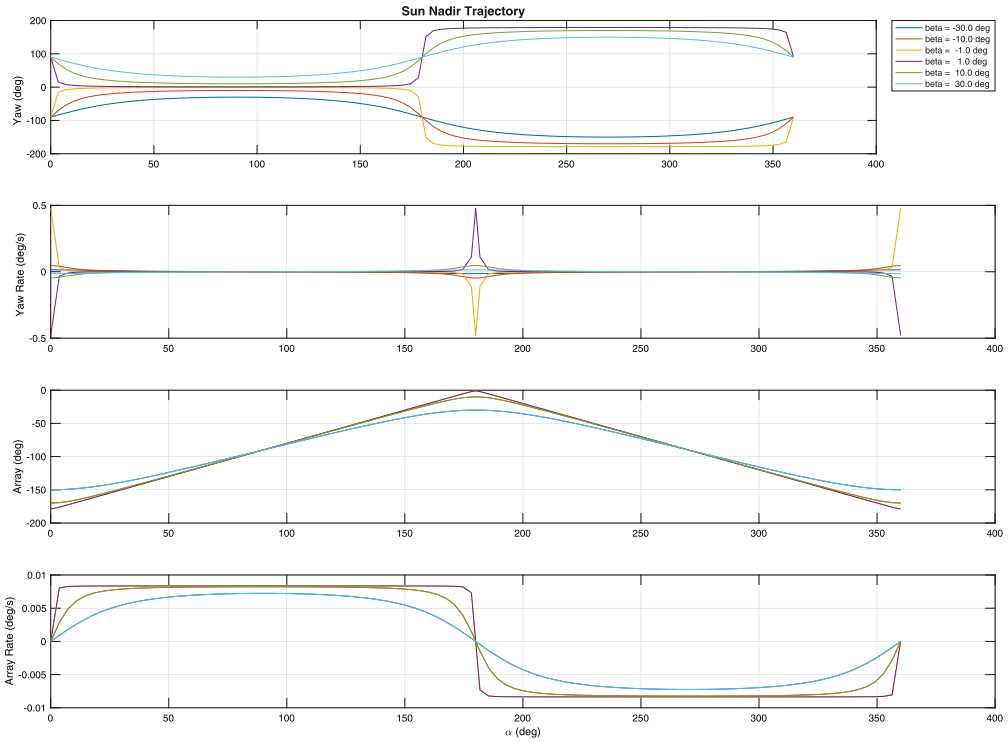


Figure 22.3 Sun-nadir yaw trajectory for a GPS orbit. The noon–midnight flips are evident.

$$1 - u_y^2 = \cos^2 \epsilon \tag{22.9}$$

$$u_y = \sin \epsilon \tag{22.10}$$

where u_y is found by transforming the Sun vector in the LVLH frame to the core frame; therefore

$$\sin\theta \sin\alpha \cos\beta - \cos\theta \sin\beta = \sin\epsilon \tag{22.11}$$

$$\theta - \theta_{ideal} = sin^{-1}\left(\frac{\epsilon}{\sin\alpha}\right) \tag{22.12}$$

The second equation is found using the relationship

$$A\sin(a+b) = a\sin a\cos b + A\cos a\sin b \tag{22.13}$$

and gives us the relationship between angular error and deviation from the ideal trajectory. This can be implemented by feeding an offset as a function of α and β into the

yaw-error measurement. When β is small this becomes

$$\theta - \theta_{ideal} = \sin^{-1}\left(\frac{\epsilon}{\sqrt{(\sin\alpha\cos\beta)^2 + \sin^2\beta}}\right) \tag{22.14}$$

or

$$\epsilon = \sin\alpha\sin(\theta - \theta_{ideal}) \tag{22.15}$$

which implies that even large errors at noon and midnight cause only small errors in the solar-array pointing.

22.5. Components

22.5.1 Sensors

Three-axis sensing is required for a Sun-nadir pointing spacecraft. One configuration is to measure roll and pitch with an Earth sensor and yaw with a Sun sensor. If the Sun sensor is array mounted, three-axis information will be available except in a narrow region about spacecraft noon and midnight. The sensing gaps can be bridged with gyros or (since as discussed above, the solar-array pointing error becomes less sensitive to yaw error as the spacecraft approaches the noon/midnight regions) we can rely on ephemeris information to estimate yaw.

Another possibility is to use a star sensor for all three axes and rely on ephemeris information to determine the orientation of the nadir vector. For very precise applications, the star sensor could be used in conjunction with gyros. For less-precise pointing, the star sensor could be used by itself. If gyros are used, a bright star sensor may be appropriate.

In this chapter, the combination of a conical scanning Earth sensor and array-mounted Sun sensors will be used. This is a relatively low-cost option and should provide adequate pointing for a GPS satellite-type application (0.1°) in roll and pitch and 3° solar-array pointing error).

Four Sun-sensor heads are mounted on the arrays, two on each array. The sensor geometry is illustrated in Fig. 22.4. One sensor's boresight is in the yz-array plane the other is in the xz-array plane. The first is canted 45 deg from y and the other 45 deg from $+z$. The elements on the other array are canted in the opposite direction so that the yz sensor points down (in the picture) and the xz sensor points to the left. The Earth sensor has a motor that rotates a mirror or prism that sweeps the field of view of the sensing element across the Earth, see Fig. 22.5. Pitch is found by measuring the center of the chord with respect to an internal reference and roll is found from the length of the chord. If the scan angle is set properly the scan will traverse the northern or southern hemisphere of the Earth resulting in a one-to-one relationship between chordwidth and attitude. This will not be the case for large attitude errors.

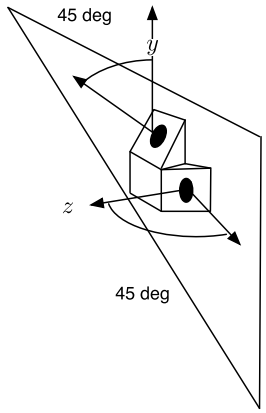


Figure 22.4 Sun-sensor geometry. These sensors are used to determine yaw. *y* is along the solar-array rotation axis. $+z$ is normal to the cell face.

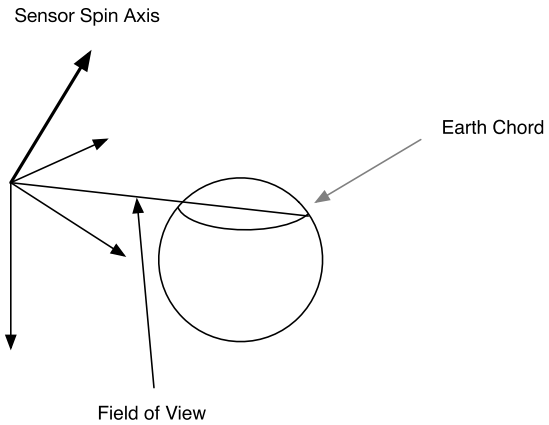


Figure 22.5 Earth-sensor geometry diagram.

22.5.2 Actuators

This design uses four reaction wheels mounted in a pyramid about the yaw axis. All four wheels run at the same time but only three are needed for three-axis control. Reaction wheels make sense in this application because the spacecraft must maneuver about both the pitch and yaw axes, and since the maneuvers are cyclic, the momentum can be transferred back and forth between the core and the reaction wheels without the use of any consumables.

Thrusters will be required to perform orbit-adjust maneuvers since misalignments and thrust mismatches in the orbit-adjust thrusters may be significant. Thrusters will also be needed for attitude control during maneuvers.

Momentum unloading can be performed by thrusters or by magnetic torquers. The torque produced by a torquer of dipole M is

$$T = M^{\times} B \tag{22.16}$$

where

$$M^{\times} = \begin{bmatrix} 0 & -M_z & M_y \\ M_z & 0 & -M_z \\ -M_y & M_z & 0 \end{bmatrix} \tag{22.17}$$

The above matrix cannot be inverted; consequently, it is not possible to realize a three-axis torque demand with magnetic torquers. However, if B changes it will be possible on average to satisfy an average three-axis demand. A simple algorithm to match the desired torque to the dipole is

$$M = -\frac{B^{\times} T}{|B|^2} \tag{22.18}$$

This is the least-squares solution to the desired torque. This requires three torquers.

An alternative is to use a dot-product algorithm. Assume we have two torquers. The torque axis of torquer i is

$$u_i = \frac{m_i \times B}{|m_i \times B|} \tag{22.19}$$

where small letters imply unit vectors. The dot product is taken between the torque demand with this vector. The torquer i is then energized to produce this torque. The dot product of the remainder of the torque is taken with the second torquer.

22.6. Attitude determination

22.6.1 Roll

A conical scanning Earth sensor returns the Earth chordwidth as the measurement. The chordwidth is related to the nadir angle, the angle between the spin axis of the sensor and the nadir vector, by the relationship

$$\cos\rho = \sin\eta \sin\gamma \cos\frac{\Omega}{2} + \cos\gamma \cos\eta \tag{22.20}$$

where r is the Earth angular radius, η is the nadir angle, Ω is the chordwidth, and γ is the cant angle between the spin-axis of the sensor and the boresight. Using the sum-of-angles trigonometric identity this becomes

$$\cos\left(\eta + \tan^{-1}\left(\tan\gamma \cos\frac{\Omega}{2}\right)\right) = \frac{\cos\rho}{\sqrt{\left(\sin\gamma \cos\frac{\Omega}{2}\right)^2 + \cos^2\gamma}} \tag{22.21}$$

The nadir angle is related to roll through the spherical triangle identity

$$\cos\left(\theta_z + \delta - \frac{\pi}{2}\right) = \cos\theta_y \cos\eta \tag{22.22}$$

where δ is the offset between the spin-axis and the spacecraft pitch-axis and θ_y is the spacecraft pitch angle.

22.6.2 Pitch

The pitch measurement is just the average of the Earth's rise and fall times with respect to a fixed reference in the sensor.

22.6.3 Sun-sensor eye preprocessing

The output from each Sun sensor eye is an analog voltage. We need to recover the value of the dot product between the Sun unit vector and the eye boresight. The response of the sensor can be written as

$$v = \sum_{k=0}^{N} a_k x^k \tag{22.23}$$

where $x = \cos\theta$. If

$$a_k << a_1 \quad k > 1 \tag{22.24}$$

then x can be extracted from v using a Newton–Raphson algorithm.

22.6.4 Solar-array pitch

The solar-array pitch measurement is computed from the Sun sensor eyes with components in the $\pm x$ directions. The algorithm is

$$\theta = \sin^{-1}\left(\frac{\Delta m}{\sqrt{2}}\right) \tag{22.25}$$

where Δm is the difference between the $+x$ and $-x$ eye measurements.

This is

$$u_y = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\theta_y \\ 0 & 1 & \theta_x \\ \theta_y & -\theta_x & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} \tag{22.26}$$

where s is the Sun vector.

22.6.5 Yaw

The measurement from the $\pm y$ eyes is of the y component of the Sun vector in the array frame. Solving for yaw in terms of u_y and s gives

$$u_y = s_y \cos\theta - s_x \sin\theta + s_z\theta_x \tag{22.27}$$

Using the sum of angles trigonometric identity

$$\theta = \sin^{-1}\left(\left(\frac{u_y - s_z\theta_x}{\sqrt{s_x^2 + s_y^2}}\right) - \tan^{-1}\left(\frac{s_y}{-s_x}\right)\right. \tag{22.28}$$

The u_y component is just

$$u_y = \frac{\Delta m}{\sqrt{2}} \tag{22.29}$$

where Δm is the difference between the $+y$ and $-y$ eyes.

Eq. (22.28) shows the sensitivity of the yaw measurement to geometry. When s_x and $s_y = 0$ the Sun lies along the z-axis and the Sun-sensor measurement is completely insensitive to yaw. When the x and y components are small, the measurement will be sensitive to yaw but the noise will be amplified due to the geometry. Consequently, there will be some regions around the spacecraft at noon and midnight where it will be difficult to get good yaw measurements from the array-mounted Sun sensors. In these regions, gyros may be used or yaw estimated from roll and spacecraft momentum.

22.7. Control

22.7.1 Reaction-wheel loop

Reaction wheels are a motor with a flywheel attached. The motor base is attached to the spacecraft. A torque applied to the spacecraft by the motor causes the wheel to rotate in the other direction. The reaction wheel will experience friction due to bearings, windage, and other sources. The friction can be modeled as

$$\begin{aligned} T_F &= -\mathrm{sgn}(\Omega)\, T_C - c\Omega \quad && |\Omega| \geq \Omega_L \\ T_F &= \left(-\left(\tfrac{T_C + c\Omega_L}{\Omega_L}\right)\right)\Omega \quad && |\Omega| < \Omega_L \end{aligned} \tag{22.30}$$

where sgn gives the sign of the argument, Ω is the wheel speed, and ω_L is the wheel speed limit below which Coulomb friction, T_C, is ignored. A pure damping model is used at low speeds to ensure that the wheel speed damps to zero in a simulation.

Otherwise, an unrealistic limit cycle will be caused by finite-size numerical integration. The reaction torque on the spacecraft is

$$T_R = T_M - T_F \tag{22.31}$$

Consequently, without knowledge of the friction there will always be a wheel-speed-dependent error. The vector angular-acceleration demand on the wheels is

$$\dot{\Omega} = \frac{K_p\theta + K_D\dot{\theta} + K_I \int \theta}{J} \tag{22.32}$$

In this example, the wheels are being used as momentum actuators since the wheels do not have a current feedback loop. The acceleration demand must be distributed among the wheels. If three orthogonal wheels are used then each component is sent to one of the three wheels. However, if more than three wheels are used it will be necessary to distribute the three acceleration components among more than three wheels. An additional relationship must be introduced. It is required that

$$\dot{\Omega} = U\dot{\Omega}_W \tag{22.33}$$

where U is a $3 \times n$ matrix, where $n \geq 3$, of the unit vectors of the reaction-wheel spin axes.

We want $\dot{\Omega}_W$ chosen so that

$$J = \frac{1}{2}\dot{\Omega}_W^T\dot{\Omega}_W + \lambda^T(\dot{\Omega} - U\dot{\Omega}_W) \tag{22.34}$$

is minimized. l is a 3-vector of Lagrange multipliers that adjoin the constraint equations to the scalar cost equation. Taking derivatives of this equation with respect to $\dot{\Omega}_W$ and λ gives the two vector equations

$$\dot{\Omega} = U\dot{\Omega}_W \tag{22.35}$$

$$\dot{\Omega}_W = U^T\lambda \tag{22.36}$$

An alternative approach is to command wheel-speed changes instead of torques. This can be done because the equations of motion (neglecting Euler coupling) are

$$I\ddot{\theta} + J(\dot{\Omega} + \dot{\omega}) = T \tag{22.37}$$

If we make

$$J\dot{\Omega} = K_p\theta + K_D\dot{\theta} + K_I \int \theta \tag{22.38}$$

then we can use tachometer loops for each wheel to control the spacecraft. The tachometer drives the wheel speed to a commanded wheel speed. The commanded wheel speed is a function of the desired attitude correction.

22.7.2 Attitude loop

Since there will be steady external disturbances on the spacecraft, the attitude loops must contain integral action. A general form for a single-axis proportional-integral-differential (PID) controller is

$$T = K\left(1 + \tau_r\left(\frac{s}{\tau_f s + 1}\right) + \frac{1}{\tau_i s}\right) \tag{22.39}$$

where K is the forward gain, τ_r is the rate time constant, τ_f is the rate filter time constant, and τ_i is the integral time constant. This is generally implemented in a digital control system. If multirate sampling is available, it may be advantageous to run the rate filter at a higher rate than the rest of the controller.

22.7.3 Solar-array control

The solar-array pitch measurements are translated into solar-array steps, and the step demand is fed to the solar-array stepping motor. The stepping motor acts like a moderately damped second-order system.

Example 22.1 shows an attitude control simulation. The momentum-management system is not included. The simulation first calls the setup script. It designs PID controllers for each axis and a PI controller for the tachometer loops. The wheels do not have current feedback so a torque demand becomes a wheel-speed demand for the wheels. The tachometer loop controls the wheel speed. Changes in wheel speed translate into changes in body rates, thus controlling the spacecraft. This can be shown in a one-dimensional example.

Assume you have a dynamical system composed of two equations for the pitch loop of a momentum-bias spacecraft.

$$T = I\ddot{\theta} + J\dot{\Omega} \tag{22.40}$$

$$T_w = J\dot{\Omega} \tag{22.41}$$

where I is the pitch inertia, J is the momentum-wheel inertia, T is an external disturbance torque, θ is the angle that you want to control, Ω is your momentum-wheel speed, and T_w is the torque on the wheel produced by an electric motor connecting the wheel to the spacecraft. You have a measurement of Ω and θ but not T_w. How do we connect the two? Write

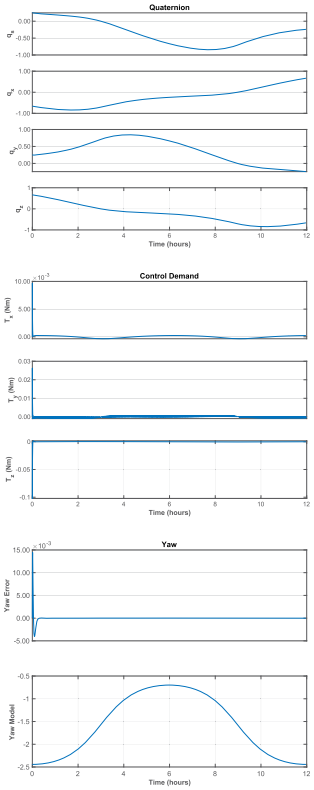
$$J\dot{\Omega} = I\left(2\zeta\sigma\dot{\theta} + \sigma^2\theta\right) \tag{22.42}$$

where ζ is the damping ratio and σ is the undamped natural frequency. Our first dynamical equation is now

$$T = I\ddot{\theta} + I\left(2\zeta\sigma\dot{\theta} + \sigma^2\theta\right) \tag{22.43}$$

which is a damped second-order system. Divide by I to clarify the equation

```
%% Run the setup script
SunNadirSetup;

%% Initial conditions
[q,angleSA,w] = ICSN( xNoSun, wo, beta, alpha );
angleSAC = [angleSA;angleSA];
x  = [q;angleSAC;w+wDelta;[0;0];[0;0;0;0]];

tW          = zeros(6,1);
roll        = 0;
pitch       = 0;
yaw         = 0;
uSunI       = [cBeta; 0; sBeta];
mSS         = SSOutput( angleSAC, nSA, uSSInSAF,
        I2B(q,uSunI), eyeSSCoeff, quantSSA );
xP          = zeros(length(x)+5,nSim);

for k = 1:nSim
 % Attitude quaternion
 qECIToBody  = x(1:4);

 % Ephemeris
 rNadir = rOrbit*[-sin(alpha);0;cos(alpha)];
 uSunB  = I2B( qECIToBody, uSunI );

 % RWA Tachometer
 wTach  = x(12:15);

 % Conical Scanning ESA
 [chordwidth, scanAngle] = ESACS( qECIToBody,
        rNadir, uESA, cantESAScan, pitchAxis );
 rollX  = CW2Roll( scanAngle, cantESAScan, uESA,
        uPitch, angEarth, chordwidth );
 [~,j]  = min(abs(rollX));
 roll   = rollX(j);
 pitch  = scanAngle;

 % Ephemeris Processing
 [yawModel,sAAngleModel,yawRateModel,saRateModel
        ] = SunNadir( xNoSun, wo, beta, alpha );

 % Solar array mounted sun sensors
 mSS = SSOutput( x(5:6), nSA, uSSInSAF, uSunB,
        eyeSSCoeff, quantSSA );
 uSunL = [ -sin(alpha)*cBeta;-sBeta;-cos(alpha)*
        cBeta ];
 [yaw,sAPitch] = SSAD( cToD*mSS, eyeSSCoeff,
        uSunL, roll );

 % The attitude control loops
 tC  = -[cRoll*xRoll   + dRoll*roll;...
         cPitch*xPitch + dPitch*pitch
         cYaw*xYaw     + dYaw*yaw];

 xRoll  =  aRoll*xRoll   + bRoll*roll;
 xPitch =  aPitch*xPitch + bPitch*pitch;
 xYaw   =  aYaw*xYaw     + bYaw*yaw;

 % Plotting
 xP(:,k) = [x;tC;yaw;yawModel];

 % Solar Array Control
 filteredSAPitch = kSAPitch*filteredSAPitch +
        (1-kSAPitch)*sAPitch;
 deltaAngle = sAStepSize*fix(filteredSAPitch/
        sAStepSize);
 angleSAC = angleSAC + [1;1]*deltaAngle;
 dRHS.eSAC = angleSAC;

 % Torque demand to RWA accel demand
 wDRWA  = -aRWA*tC;

 % Integrate to get wheel speed demand
 wRWAC  = wRWAC + dT*wDRWA;

 % The RWA Tachometer Loops
 wError =  wTach - wRWAC;
 tRWA   = -dTL*wError - cTL*xTL;
 xTL    =  xTL + aTL*xTL + bTL*wError;

 dRHS.tW(iRWA) = tRWA;
 x          = RK4(@RHSSNP,x,dT,0,dRHS);

 % Update the orbit angle
 alpha  = alpha + wo*dT;
end

%% Plotting
t = (0:nSim-1)*dT;
TimeHistory(t,xP(1:4,:),{'q_s';'q_x';'q_y';'q_z'
        },'Quaternion');
TimeHistory(t,xP(7:9,:),{'\omega_x';'\omega_y';'\
        omega_z'},'Body Rates');
TimeHistory(t,xP(12:15,:),{'\omega_1'...
'\omega_2'  '\omega_3' '\omega_4'}, 'Wheels');
TimeHistory(t,xP(16:18,:),{'T_x (Nm)' 'T_y (Nm)'
        'T_z (Nm)'},'Control Demand');
TimeHistory(t,xP(19:20,:),{'Err' 'Model'},'Yaw');
```



Example 22.1: Attitude control system simulation.

$$\frac{T}{I} = \ddot{\theta} + \zeta\sigma\dot{\theta} + \sigma^2\theta \tag{22.44}$$

Our inner loop commands wheel speed.

$$T_w = Jk(\Omega_c - \Omega) \tag{22.45}$$

where k is the gain and Ω_c is the commanded wheel speed. The inner-loop dynamical equation is now

$$k\Omega_c = \dot{\Omega} + k\Omega \tag{22.46}$$

which is a damped first-order system. If k is high enough, that is the inner loop is very fast, the wheel speed will equal the commanded speed.

$$\Omega = \Omega_c \tag{22.47}$$

This is the "steady-state" response. How do we determine Ω? Integrate Eq. (22.42)

$$\Omega = \left(\frac{I}{J}\right)\int\left(2\zeta\sigma\dot{\theta} + \sigma^2\theta\right) \tag{22.48}$$

or

$$\Omega_c = \left(\frac{I}{J}\right)\sigma\left(2\zeta\dot{\theta} + \sigma\int\theta\right) \tag{22.49}$$

Our outer loop will control θ. Our inner loop will control Ω.

The function ICSN initializes the ECI to body quaternion and the solar-wing angles. β is the elevation of the Sun with respect to the orbit plane and α is the azimuth. The wing-mounted Sun sensors are modeled using SSOutput and SSAD. The conical scanning Earth sensor uses CW2Roll and ESACS.

Solar-array control uses a stepping motor. A first-order filter is applied to the pitch angle. The effect of the stepping motor can be seen in the y-axis plot.

22.7.4 Momentum control

The spacecraft is nominally zero momentum in the absence of external disturbances. Of course, external disturbances will cause the momentum to grow and this momentum growth must be controlled. The reaction wheels can only exchange momentum with the body; hence some other type of actuator is required to remove momentum. The obvious choices are either thrusters or magnetic torquers. Magnetic torquers have the advantage that they do not produce a force on the spacecraft and therefore do not perturb the orbit, a major advantage if the satellite is used for navigation purposes. In addition, magnetic torquers can only remove momentum about two axes at one time although on average, if the magnetic-field direction in the body frame changes, they can remove momentum in all three axes.

The momentum in the spacecraft in the inertial frame is

$$H = A\left(I\omega + \sum_{i=1}^{N} u_i J(\Omega_i + u_i^T \omega)\right) \tag{22.50}$$

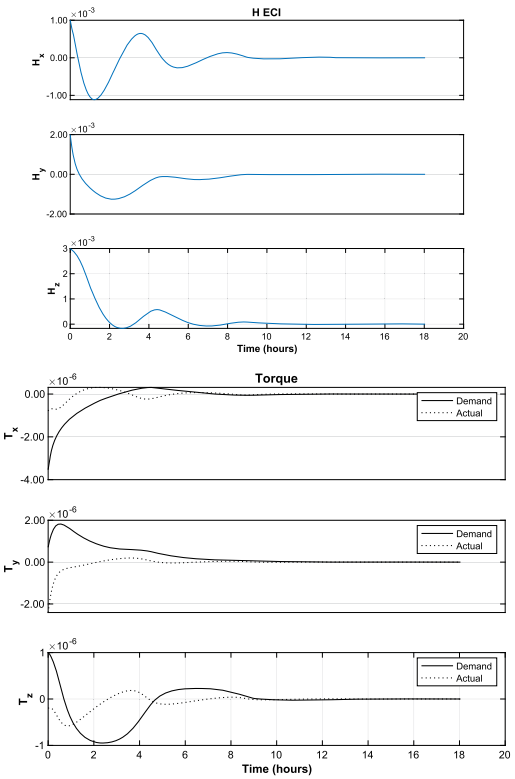
where ω is the body rate, Ω_i is the angular rate of the wheel with respect to the spacecraft, and u_i is the spin-axis of the ith wheel. The momentum control law is a proportional controller

$$T = -KH \tag{22.51}$$

Momentum unloading is shown in Example 22.2. The simulation uses Euler integration where

$$\dot{x} = f(x, u) \tag{22.52}$$

```
%% Momentum Unloading using two torquers.
% The spacecraft is aligned with LVLH.
% The torquers are along roll and yaw.

el      = [22000 pi/4 0 0 0 0];
p       = Period(el(1));
t       = linspace(0,2*p,1000);
[r,v ]  = RVOrbGen(el,t);
qLVLH   = QLVLH(r,v);
jD0     = Date2JD([2024 8 1]);
jD      = jD0 + t/86400;
dT      = t(2);
K       = 0.001;

bECI    = BDipole(r,jD);
bLVLH   = QForm(qLVLH,bECI);
u1      = [1;0;0];
u2      = [0;0;1];
n       = length(t);
xP      = zeros(9,n);

hECI    = [1;2;3]*1e-3;

% Simulate using Euler integration
for k = 1:n
  torque  = -K*hECI;
  torque  = QForm(qLVLH(:,k),torque);
  um1     = Unit(Cross(u1,bLVLH(:,k)));
  um2     = Unit(Cross(u2,bLVLH(:,k)));
  torq1   = um1'*torque*um1;
  torq2   = um2'*(torque-torq1)*um2;
  torqECI = QTForm(qLVLH(:,k),torq1+torq2);
  xP(:,k) = [hECI;torque;torqECI];
  hECI    = hECI + dT*torqECI;
end

yL = {'H_x' 'H_y' 'H_z' 'T_x' 'T_y' 'T_z'};
iL = {{'Demand','Actual'}
      {'Demand','Actual'}
      {'Demand','Actual'}};
k = 1:3;
TimeHistory(t,xP(k,:),yL(k),'H ECI');
k = 4:9;
TimeHistory(t,xP(k,:),yL(4:6),'Torque',...
  {[1 4] [2 5] [3 6]},iL);
```



Example 22.2: Momentum unloading with two torquers in a 22 000-km orbit.

is replaced by

$$x_{k+1} = x_k + \Delta t f(x_k, u_k) \tag{22.53}$$

where δt is the time step.

CHAPTER 23

Lander control

23.1. Space story

I was in the Soviet Union during the summer of 1969 near Leningrad to study Russian. Apollo 11 landed on the moon and we watched it on Soviet TV. Several of my fellow students told me how people in the street came up to them and, upon discovering that they were American, offered their congratulations on the successful landing.

23.2. Landers

The first landers, on any body except the Earth, were the lunar landers in the 1960s. Surveyor was the first US robotic lander. This was followed by the Lunar Module, which had a pilot. The Surveyor and Lunar Module landing trajectories are shown in Fig. 23.1.

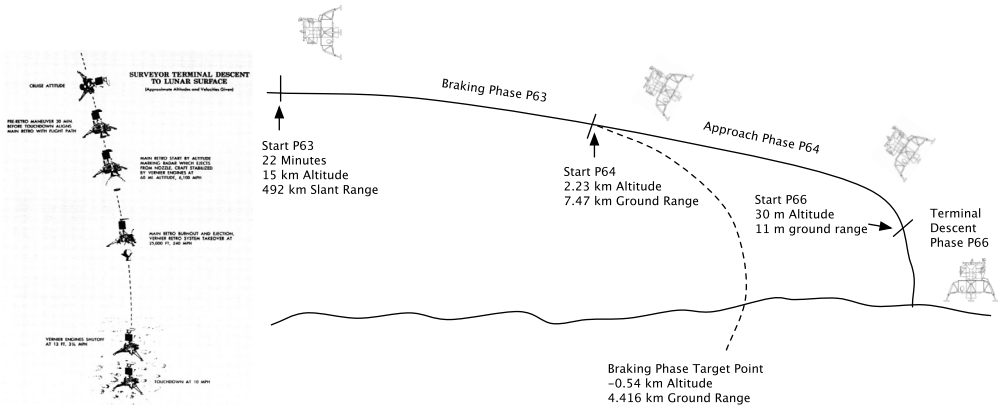


Figure 23.1 Surveyor (on the left) and Lunar Module landing trajectories (on the right). Images courtesy of NASA.

Entry and landing on planets or moons with an atmosphere usually involve atmospheric braking and parachutes for part of the descent. The NASA JPL Mars landers use a system shown in Fig. 23.2.

https://doi.org/10.1016/B978-0-32-399915-1.00035-8

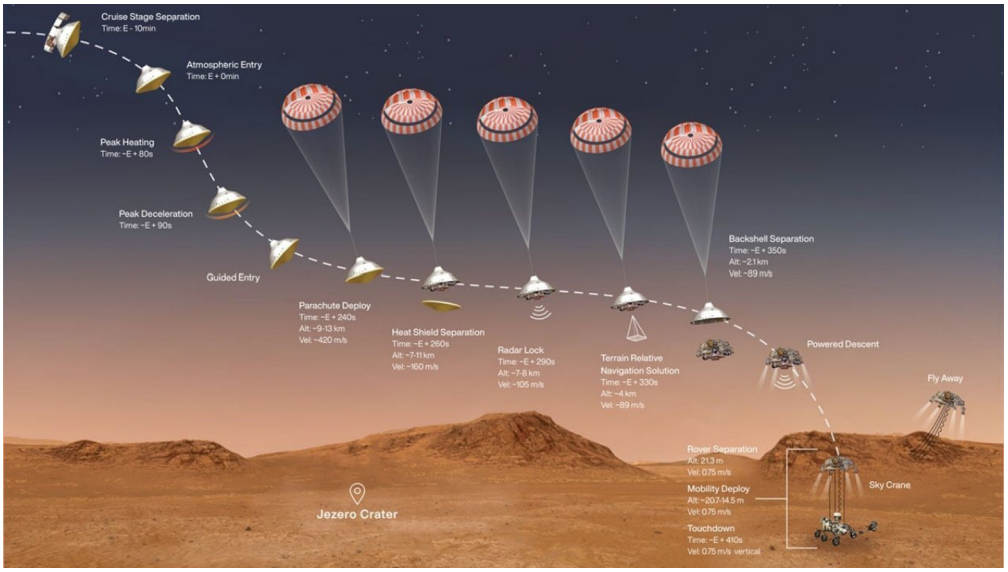


Figure 23.2 NASA JPL Mars lander. Image courtesy of NASA.

23.3. Landing concept of operations

In this chapter, we will focus on powered descent. This is the type of landing done on the Moon or other airless moons. The concept of operations (CONOPS) is shown in Fig. 23.3. The mission sequence has the following stages

1. Reorientation to the burn descent orientation;
2. Powered descent to the terminal point;
3. Hover at the terminal point to determine a safe landing spot;
4. Translation to a safe spot;
5. Vertical descent;
6. Touchdown and engine shutdown.

The landing sequence starts in a near-circular orbit near the surface. The parking orbit is assumed to be over the landing spot. Since the Moon rotates this is only true for a short period. The guidance algorithm needs to account for the surface movement relative to the inertial frame.

In this chapter, we will use a lunar lander as an example.

23.4. Selenographic coordinates

The selenographic frame is shown in Fig. 23.4. ϕ'_m is the Selenographic Latitude, which is the acute angle measured normal to the Moon's equator between the equator and

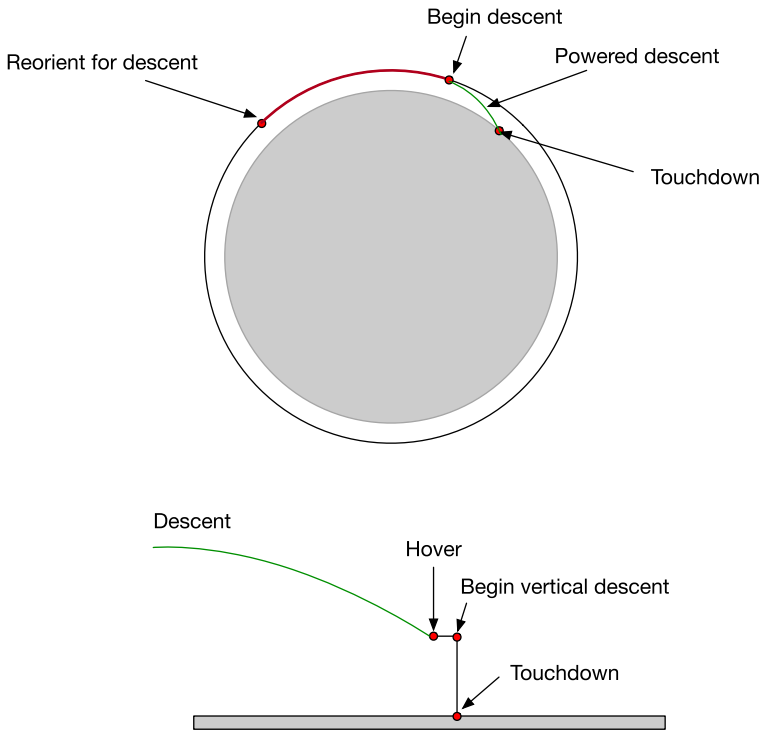


Figure 23.3 Lunar-landing concept of operations.

a line connecting the geometrical center of the coordinate system with a point on the surface of the Moon. The angle range is between -90 and +90 degrees. λ_m is the Selenographic Longitude, which is the angle measured towards the West in the Moon's equatorial plane, from the lunar prime meridian to the object's meridian. The angle range is between 0 and 360 degrees. North is the section above the lunar equator containing Mare Serenitatis. West is measured towards Mare Crisium.

The relationship to the ECI frame is shown in Fig. 23.5. i is the inclination between the Moon's equator and the ECI equator. Ω' is the longitude of the mean ascending node of the lunar orbit referenced to the mean equinox. ω is the angle between the ascending node and the lunar prime meridian.

The linear tangent guidance law is an analytical solution to the powered-descent problem assuming constant acceleration [1]. Since the mass of the lander decreases with time this assumes a variable thrust. The problem is posed as a two-dimensional flat planet problem with a uniform vertical gravity field. This is the same problem solved in [2].

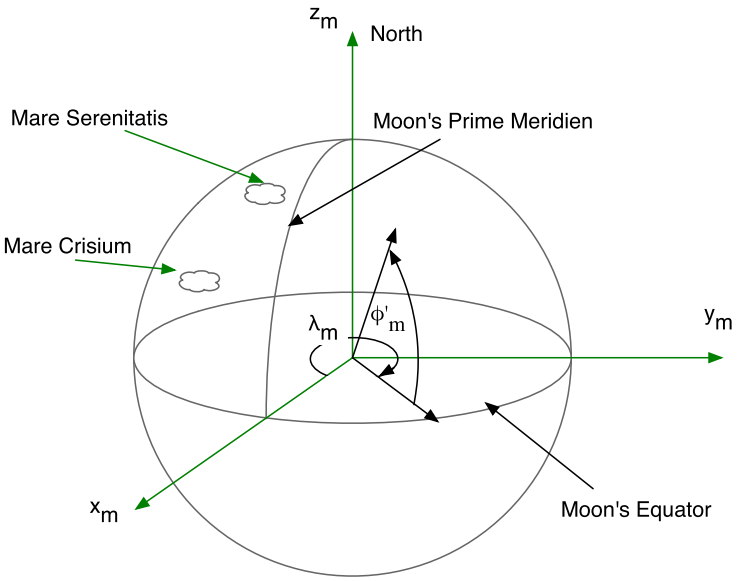


Figure 23.4 Selenographic frame.

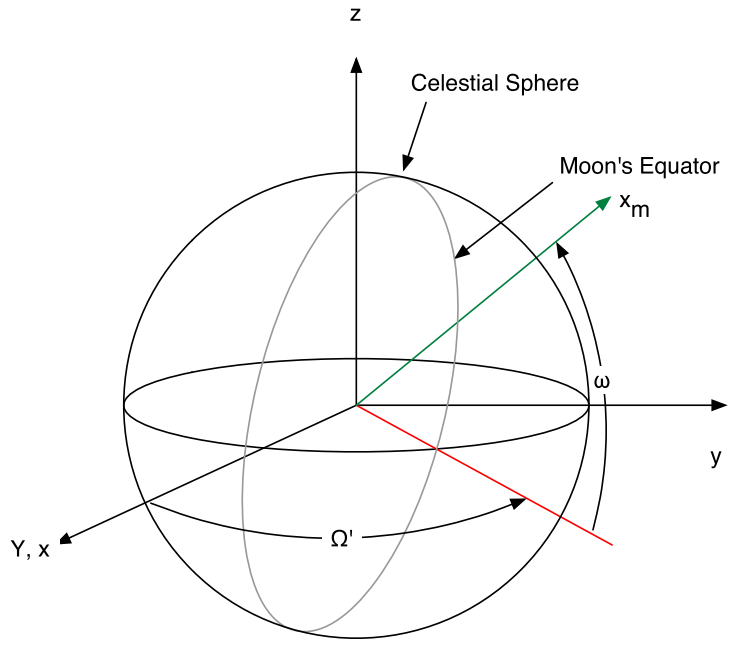


Figure 23.5 Selenographic to ECI frame.

23.5. Linear-tangent guidance law

The linear-tangent guidance law is the solution to the problem

$$\dot{s} = f(s, \beta) \tag{23.1}$$

where the state vector is

$$s = \begin{bmatrix} x \\ y \\ u \\ v \end{bmatrix} \tag{23.2}$$

and the right-hand side is

$$f(s, \beta) = \begin{bmatrix} u \\ v \\ a\cos\beta \\ a\sin\beta - g \end{bmatrix} \tag{23.3}$$

where u is the horizontal velocity along x, v is the vertical velocity along y, β is the thrust direction angle, a is the rocket acceleration, and g is the acceleration of gravity in the $-y$ direction. a is assumed constant and the only control is β.

We want to solve the minimum-time descent problem. We do not care about the final value of x. Once we solve the problem, we make sure we start the final x distance from the landing point. The boundary conditions are

$$s(0) = \begin{bmatrix} 0 \\ h \\ U \\ 0 \end{bmatrix} \tag{23.4}$$

$$s(t_f) = \begin{bmatrix} \sim \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{23.5}$$

where $x(T)$ is not specified. We write the Hamiltonian as

$$H = L(s, \beta, t) + \lambda^T(t) f(s, \beta, t) \tag{23.6}$$

The control is found from

$$\frac{\partial H}{\partial \beta} = 0 \tag{23.7}$$

The cost is

$$J = \phi\left[s(t_f), t_f\right] + \int_0^{t_f} \left[H - \lambda^T f\right] dt \tag{23.8}$$

In our problem $L = 1$ and $\phi = 0$ so $J = t_f$. This makes the cost the time of flight.

The minimum-time problem is solving the two-point boundary value problem using the following state and costate equations

$$\dot{s} = f(s, \beta) \tag{23.9}$$

$$\dot{\lambda} = -\left(\frac{\partial f}{\partial s}\right)^T \lambda \tag{23.10}$$

$$0 = \left(\frac{\partial f}{\partial \beta}\right)^T \lambda \tag{23.11}$$

and the costate vector is

$$\lambda = \begin{bmatrix} \lambda_x \\ \lambda_y \\ \lambda_u \\ \lambda_v \end{bmatrix} \tag{23.12}$$

Three states are specified at the end. Since one is not specified, the corresponding costate must be zero when $t = t_f$. Thus the minimum time conditions are

$$\lambda_x(t_f) = 0 \tag{23.13}$$

$$\left(\lambda^T f\right)_{t_f} = -1 \tag{23.14}$$

The costate equations are

$$\dot{\lambda}_x = 0 \tag{23.15}$$

$$\dot{\lambda}_y = 0 \tag{23.16}$$

$$\dot{\lambda}_u = -\lambda_x \tag{23.17}$$

$$\dot{\lambda}_v = -\lambda_y \tag{23.18}$$

The control condition is

$$0 = \lambda_v \cos\beta - \lambda_u \sin\beta \tag{23.19}$$

Since λ_x is zero at $t = t_f$ it is always zero. λ_u is a constant that we can set to 1. Therefore the control is the linear tangent law

$$\tan\beta = \tan\beta_0 - ct \tag{23.20}$$

The problem is finding β_0, c, and t_f. These are found by solving the differential equations for u, v, and γ. The differential equations are most easily solved by replacing t with β as the independent variable. Taking the derivative of the linear tangent law with respect to β gives

$$dt = -\frac{\sec^2\beta}{c}d\beta \tag{23.21}$$

The equations for u, v, and γ are now

$$\frac{du}{d\beta} = -\frac{a}{c}\sin\beta\sec^2\beta = -\frac{a}{c}\sec\beta \tag{23.22}$$

$$\frac{dv}{d\beta} = -\frac{a}{c}\sin\beta\sec^2\beta = -\frac{a}{c}\sec\beta\tan\beta \tag{23.23}$$

$$\frac{d\gamma}{d\beta} = -\frac{v}{c}\sec^2\beta \tag{23.24}$$

The effect of gravity is to add the particular solutions $-gt_f$ to the v solution and $-\frac{1}{2}gt_f^2$ to the γ solution. This gives us three equations in the three unknowns of β_0, t_f, and c. The solution for v is

$$v = \frac{a}{c}(\sec\beta_0 - \sec\beta) \tag{23.25}$$

The third equation becomes

$$\frac{d\gamma}{d\beta} = -\frac{a}{c^2}\sec^3\beta - \left(\frac{a}{c^2}\sec\beta_0 + gt_f\right)\sec^2\beta \tag{23.26}$$

The three equations that need to be solved for the three constants are

$$\frac{Uc}{a} = \log|\sec\beta_f + \tan\beta_f| - \log|\sec\beta_0 + \tan\beta_0| \tag{23.27}$$

$$0 = \frac{a}{c}\left(\sec\beta_0 - \sec\beta_f\right) - gt_f \tag{23.28}$$

$$\begin{aligned} h = {} & \left(\frac{a}{c^2}\sec\beta_0 + gt_f\right)\left(\tan\beta_0 - \tan\beta_f\right) \\ & + \frac{a}{2c^2}\left(\sec\beta_0\tan\beta_0 + \log|\sec\beta_0 + \tan\beta_0| - \sec\beta_f\tan\beta_f\right. \\ & \left. - \log|\sec\beta_f + \tan\beta_f|\right) - \frac{1}{2}gt_f^2 \end{aligned} \tag{23.29}$$

where

$$\tan\beta_f = \tan\beta_0 - ct_f \tag{23.30}$$

These cannot be solved analytically and must are solved using downhill simplex, fminsearch, in MATLAB®.

23.6. Lunar-lander model

A six-degrees-of-freedom model was built for training and testing terrain-relative navigation. It includes the rotational and translational degrees-of-freedom. The dynamical model is

$$T_t = I\dot{\omega} + \dot{I}\omega + \omega^{\times} I\omega \tag{23.31}$$

$$F_g + F_t = m\dot{v} \tag{23.32}$$

$$\dot{r} = v \tag{23.33}$$

$$\dot{m} = -\frac{F_t}{u_e} \tag{23.34}$$

$$\dot{I} = \dot{m} I_s \tag{23.35}$$

$$\dot{q} = g(q, \omega) \tag{23.36}$$

$$I_s = \frac{I(0)}{m(0)} \tag{23.37}$$

where q is the attitude quaternion, I is the inertia matrix, F_g is the gravitational force, F_t is the thruster force, m is the mass, u_e is the engine-exhaust velocity, T_t is the thruster torque, and ω is the angular rate. I_s is a constant scaling matrix that relates inertia to mass. Slosh is not modeled as the tanks are assumed to have baffles that damp fuel motion.

Fig. 23.6 gives images of landers. Many commercial companies and government organizations are also building and flying landers. The most recent, at the time of writing this book, is the Chinese Chang'E-5 probe that sent samples back to Earth.

The lunar-lander model is given in Table 23.1. We assume a bipropellant engine. The fuel load is computed from the rocket equation, modified to include tankage fraction.

Table 23.1 Lander parameters.

Parameter	Symbol	Value
Dry mass	$m_p + fm_f$	22 kg
Fuel mass	m_f	20.4 kg
Fuel-tankage fraction	f	0.1
Main engine-exhaust velocity	u_e	325 s
Descent acceleration multiple	n	3
Descent time	t_d	365.8 s
Hover time	t_h	120 s
Hover altitude	h_h	100 m

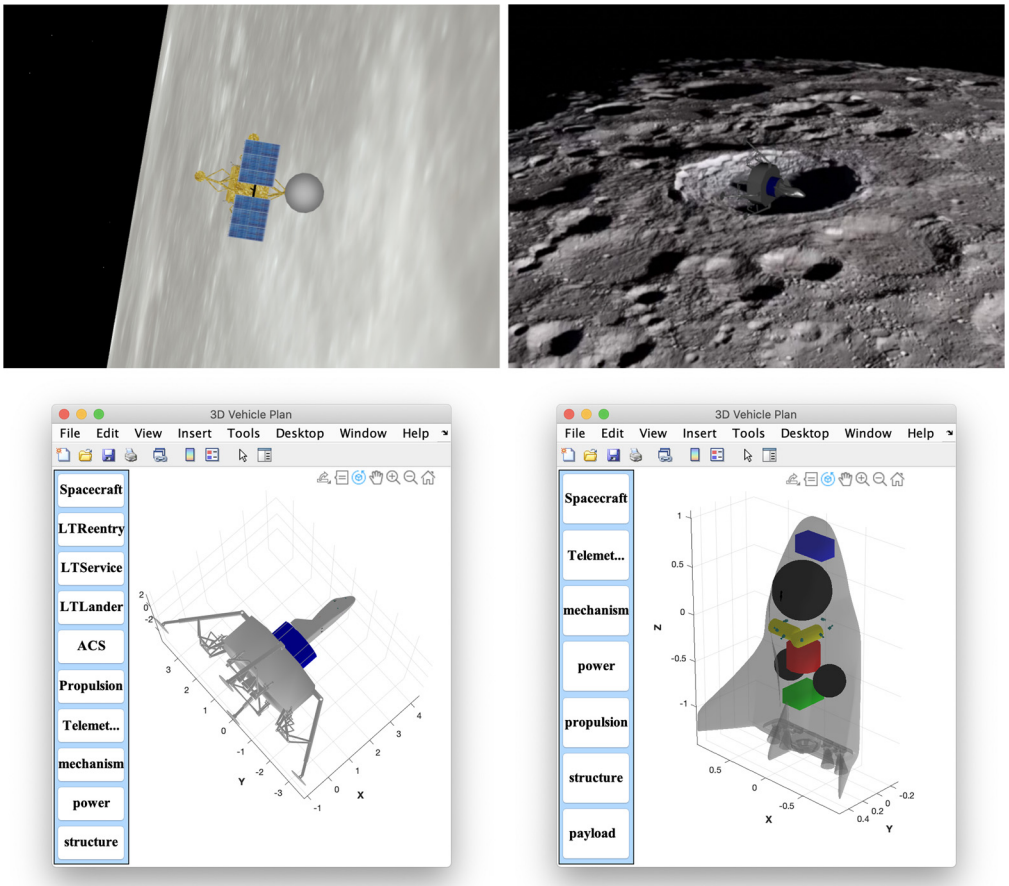


Figure 23.6 Proposed landers. The upper left is a lander with a high-gain antenna. The next three are a proposed commercial lunar helium-3 return vehicle. The reentry vehicle would be reusable. All of the avionics are in the reentry vehicle, include all RCS thrusters. Images courtesy of Princeton Satellite Systems.

$$\gamma = e^{\frac{\Delta u}{u_e}} \tag{23.38}$$
$$m_f = m_p \frac{\gamma - 1}{1 + (1 - \gamma)f}$$

where m_p is the mass of everything except the tanks and fuel, and f is the fuel-tankage fraction.

The total velocity change includes all the terms in Table 23.2.

This results in a dry mass of 22 kg and a fuel mass of 20.4 kg.

Table 23.2 Landing-velocity change contributions. g is the acceleration of lunar gravity, n is the descent control acceleration multiple, t_d is the descent time t_h is the over time, t_a is the bang-bang descent time, and t_s is the bang-band switch time.

Contribution	Equation	Value (m/s)
Linear tangent descent	ngt_d	1781
Hover	gt_h	194
Landing	ngt_a	16
Attitude control	5% of Δu fuel load	100
Total		2091

23.7. Optimal descent

In this chapter, we will use a single optimal solution for the powered descent. The descent will assume a constant deceleration at three times the acceleration of the lunar gravity. Example 23.1 shows the optimal trajectory. The first pair of plots shows the linear tangent law. The second pair shows the "optimal" trajectory using fmincon. The terminal altitude is 100 m where the hover will commence.

23.8. Descent control

Descent control requires that the thrust vector be aligned with the desired inertial vector. In addition, it is usually necessary that a landing sensor, such as radar, a navigation camera, or lidar, be pointed at the lunar surface. The required attitude quaternion is found in the two vectors. Assume that the sensor vector is the trajectory radius vector, n, and the thrust vector is u. Both vectors are defined in the ECI frame. Assume that the thrust vector is along x in the body frame. The quaternion must align x with u and bring the sensor vector, in the body frame, b, close to n. Fig. 23.7 shows the geometry.

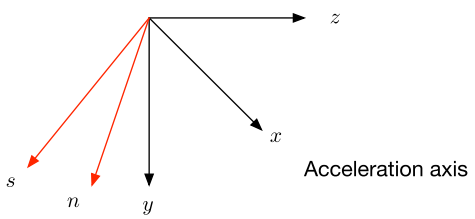
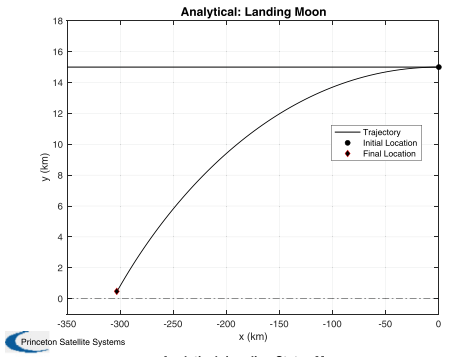
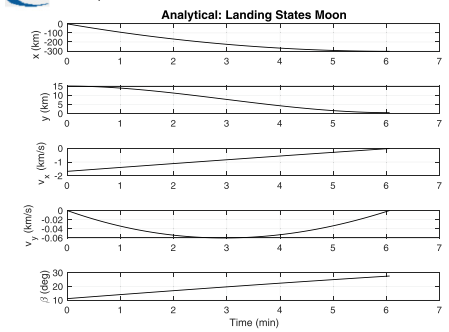
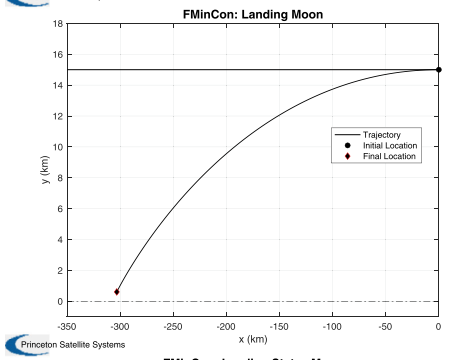


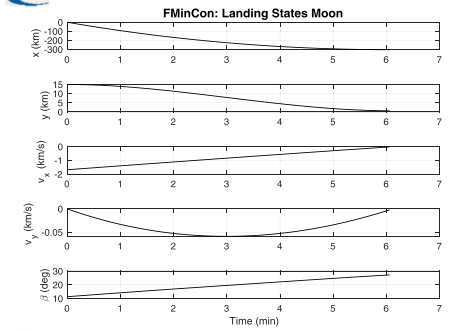
Figure 23.7 Aligning two vectors with ECI vectors. The acceleration vector is aligned exactly, the other as close as possible.

```
1
2
3 %% Parameters
4 muMoon          = Constant('mu moon');
5 rMoon           = Constant('equatorial radius moon'
      );
6 d               = struct;
7 d.u             = sqrt(muMoon/rMoon); % Orbit
      velocity
8 d.g             = muMoon/rMoon^2; % Gravity
9 d.h             = 15; % Altitude (km)
10 d.n            = 100; % Number of increments
11 d.a            = 3*d.g; % Engine acceleration
12 d.x            = [0;d.h;-d.u;0]; % Initial state
13 algorithm      = 'interior-point';
14
15 % Find the thrust direction angles
16 [beta, t, tMin]   = BilinearTangentLaw( d.u, d.g,
       d.a, d.h, d.n );
17 fprintf(1,'Analytical minimum time %12.4s sec\n',
       tMin);
18
19 % We don't want the last beta since the vehicle
       is on the ground
20 beta         = beta(1:end-1);
21 d.n          = d.n-1;
22
23 % Do this to get a landing
24 beta         = fliplr(beta);
25
26 % Simulate the landing
27 Simulate2DLanding( t, beta, d, 'Analytical' );
28
29 %% Now repeat with fmincon
30
31 % fmincon options
32 opts = optimset( 'Display','iter-detailed',...
33                  'TolFun',0.6,...
34                  'algorithm',algorithm,...
35                  'TolCon',1e-5,...
36                  'MaxFunEvals',100000);
37
38 d.hFinal = 0.6;
39
40 % The cost is time, which is a decision variable
41 % The cost is the time to reach the final state
       vector
42 costFun    = @(x) LandingCost2D(x,d);
43
44 % The numerical integration of the state is in
       the constraint function
45 constFun         = @(x) LandingConst2D(x,d);
46
47 % The final state vector is [x;0;0;0];
48 % We don't care what x is since we can always
       start the descent at the
49 % appropriate time.
50
51 % First guess for the time decision variable
52 dT         = t(2:end) - t(1:end-1);
53
54 % Do this to get a reasonable first guess but not
       exact
55 beta       = 1.2*beta;
56
57 % The decision variables are acceleration angle
       and time increment
58 x0         = [beta';dT'];
59
60 % Lower and upper bounds
61 lB         = zeros(length(x0),1);
62 uB         = [(pi/2)*ones(length(dT),1);100*ones(
       length(dT),1)];
63
64 % Find the optimal decision variables.
65 x          = fmincon(costFun,x0,[],[],[],[],lB,uB,
       constFun,opts);
66
67 %% Set up the simulation
68 beta       = x(1:d.n)';
69 dT         = x(d.n+1:2*d.n);
70 t          = zeros(1,length(dT));
71 for k = 2:d.n+1
72   t(k) = t(k-1) + dT(k-1);
73 end
74
75 % Simulate the landing
76 Simulate2DLanding( t, beta, d, 'FMinCon' );
77
78 a = d.a;
79 h = d.h;
80 tDescend = t(end);
81 save('LandingTrajectory','beta','t','h','a','
       tDescend','tDescend');
```









Example 23.1: Lunar descent.

Let a be the acceleration unit vector.

$$z = a \times s \tag{23.39}$$

$$y = z \times a \tag{23.40}$$

$$m = \begin{bmatrix} x^T \\ y^T \\ z^T \end{bmatrix} \tag{23.41}$$

$$\theta = \tan^{-1} \frac{z^T n}{y^T n} \tag{23.42}$$

$$q = \begin{bmatrix} \cos \frac{\theta}{2} \\ a_x \sin \frac{\theta}{2} \\ a_y \sin \frac{\theta}{2} \\ a_z \sin \frac{\theta}{2} \end{bmatrix} \tag{23.43}$$

23.9. Terminal control

The terminal control uses a bang-bang translation controller for vertical control during terminal ascent. Horizontal control is via velocity damping. It is assumed that the lander is hovering above the landing spot.

Thrust-vector orientation is achieved by reorienting the vehicle or the engine nozzle. The reorientation controller needs to be fast enough so that the engine is pointing along the desired vector. It can set the throttle to zero until you reach the desired orientation.

The terminal descent control has four modes:

1. Mode 1 - Drive the vertical velocity to zero and hover;
2. Mode 2 - Drive all velocities to zero;
3. Mode 3 - Bang-bang vertical control to land;
4. Mode 4 - Engine shuts down.

Once it reaches Mode 4 the control system needs to set the throttle to zero. The first two are velocity-nulling controllers. The altitude-hold controller is used for both.

23.10. Altitude hold

The altitude-hold algorithm is a proportional-derivative (PD) controller in position and velocity. The control algorithms are

$$\epsilon = r - r_{set} \tag{23.44}$$

$$f = mK_h \begin{bmatrix} -\epsilon_x - \tau_h v_x \\ -\epsilon_y - \tau_h v_y \\ \epsilon_z + \tau_h v_z - g \end{bmatrix} \tag{23.45}$$

where m is the mass, g is the acceleration of gravity, aligned with z, K_h is the forward gain, τ_h is the damping time constant, and r_{set} is the position set point. This allows the vehicle to maneuver in three dimensions as needed. The thrust is

$$T = \frac{f}{|f|} \tag{23.46}$$

The thrust vector in the body frame is

$$T_V = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{23.47}$$

If q is the quaternion that rotates T_V into f, then

$$u = q \otimes \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{23.48}$$

which is the vector that the attitude control system needs to track. Example 23.2 shows the altitude-hold control system.

23.11. Bang-bang landing algorithm

Given the gravitational acceleration, g, the position and velocity at time t_s are

$$z = h - \frac{1}{2} g t_s^2 \tag{23.49}$$

$$v = g t_s \tag{23.50}$$

where t_s is the switch time and h is the initial altitude. Given the rocket acceleration a, the time to reach zero velocity is

$$t_a = \frac{g}{a} t_s \tag{23.51}$$

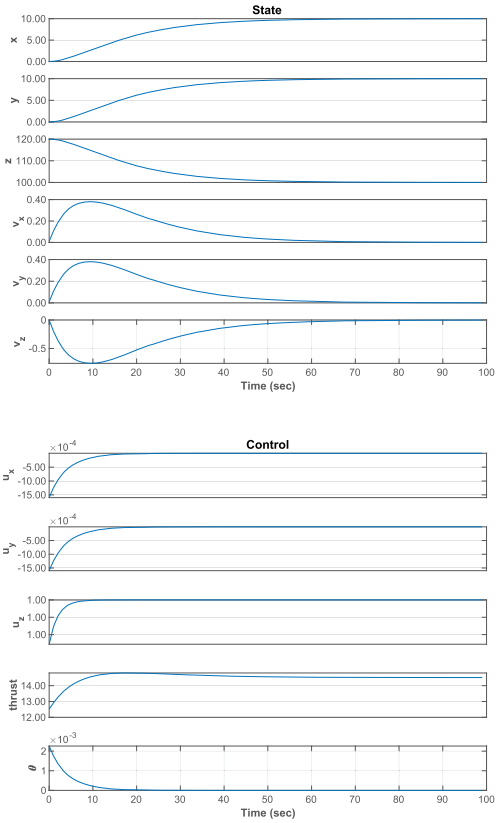
The position must be zero at time t_a

$$h - \frac{1}{2} g t_s^2 - g t_s t_a + \frac{1}{2} a t_a^2 = 0 \tag{23.52}$$

We can then solve for t_s

$$t_s = \sqrt{\frac{2h}{g(1 + \frac{g}{a})}} \tag{23.53}$$

```
mu = 4.9028e+03;
rM = 1738;
h  = 100;
dT = 1;
n  = 100;
g  = 1000*mu/(rM+h)^2;
xP = zeros(11,n);
m  = 10;
r  = 1.2*h;
x  = [0;0;r;0.01;0.01;0];
xSet = [10;10;h];
omegaN = 0.1;
tauH   = 2/omegaN;
kH     = omegaN^2;
thrust = 400;
for k = 1:n
  [u,thrust,angle] = AltitudeHold(x,kH,tauH,g,m,
        xSet);
  xP(:,k)     = [x;u;thrust;angle];
  x           = RK4(@RHS,x,dT,0,u*thrust/m,g);
end

t   = (0:n-1)*dT;
yL  = {'x' 'y' 'z' 'v_x' 'v_y' 'v_z'};
TimeHistory(t,xP(1:6,:),yL,'State')
TimeHistory(t,xP(7:11,:),{'u_x' 'u_y' 'u_z' '
      thrust' '\theta'},'Control')

%% Orbit equations
function xDot = RHS(x,~,accel,g)

r    = x(1:3);
xDot = [x(4:6);accel-[0;0;g]];

end
```



Example 23.2: Attitude-hold controller performance. It moves the vehicle 10 meters in x and y while maintaining altitude, given a small initial altitude error.

This requires accurate knowledge of the thrust. This would be obtained from the accelerometers on the IMU. The velocity change is

$$\Delta u = at_a \tag{23.54}$$

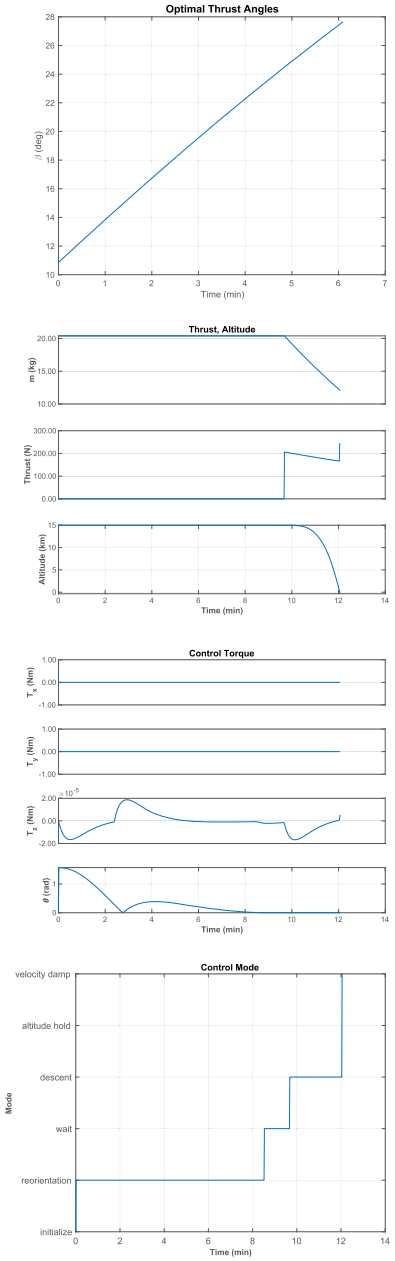
23.12. Simulation results

Example 23.3 simulates the lander control system.

References

[1] A.E. Bryson, Y.-C. Ho, Applied Optimal Control, Hemisphere, 1975.
[2] B. Acikmese, S.R. Ploen, Convex programming approach to powered descent guidance for Mars landing, Journal of Guidance, Control, and Dynamics 30 (5) (2007) 1353–1366.

```
% Constants
secInDay          = 86400;
rMoon             = 1738;
mu                = 4.9028e+03;

%% User inputs
dT                = 1; % sec
n                 = 10000; % Number of steps
startDate         = [2023 5 1 0 0 0];
fuelMass          = 20.4;
dryMass           = 22;
uE                = 325*9.806;

%% Set up the simulation
t                 = (0:n-1)*dT;

% Set up the data structures
dRHS              = RHSLunarLander;
dRHS.uE           = uE;
dRHS.massDry      = dryMass;
dLLC              = LunarLandingControl;
dLLC.traj         = load('LandingTrajectory.mat');

% Initial state and Julian date
dRHS.jD0          = Date2JD(startDate);
dLLC.jDTouchdown = dRHS.jD0 ...
+  (dLLC.tReorientation+dLLC.traj.tDescend + 120)
       /secInDay;
r   = rMoon + dLLC.traj.h;
rV  = [r;0;0;sqrt(mu/r);0];
x   = [rV;dRHS.x0(7:end)];
x(14) = fuelMass;
jD = dRHS.jD0 + t/secInDay;

%% Simulation loop
xP = zeros(length(x)+7,n);
for k = 1:n
  [dLLC, dRHS] = LunarLandingControl( x, t(k),
        dRHS, dLLC );
  alt = Mag(x(1:3))-rMoon;
  xP(:,k) = [x;dRHS.thrust;alt;dRHS.torque;...
  dLLC.angError;dLLC.modeNumber];
  x   = RK4(@RHSLunarLander,x,dT,t(k),dRHS);
  if( alt <= 0 )
    break
  end
end

%% Plot
t = t(1:k);
xP = xP(:,1:k);
yL = [dRHS.states(:)' {'Thrust (N)'} ...
{'Altitude (km)'}...
{'T_x (Nm)'} {'T_y (Nm)'} {'T_z (Nm)'}...
{'\theta (rad)'} {'Mode'} ];

% Plots
k =  1:3;
TimeHistory(t,xP(k,:),yL(k),'Position');
k =  4: 6;
TimeHistory(t,xP(k,:),yL(k),'Velocity');
k =  7:10;
TimeHistory(t,xP(k,:),yL(k),'Quaternion');
k = 11:13;
TimeHistory(t,xP(k,:),yL(k),'Angular Velocity');
k = [14 21 22];
TimeHistory(t,xP(k,:),yL(k),'Thrust, Altitude');
k = 15:20;
TimeHistory(t,xP(k,:),yL(k),'Inertia');
k = 23:26;
TimeHistory(t,xP(k,:),yL(k),'Control Torque');
k = 27;

% Control mode display
TimeHistory(t,xP(k,:),yL(k),'Control Mode');
s = {'initialize','reorientation','wait' 'descent
    ' ,...
  'altitude hold' 'velocity damp' 'landing','
       shutdown'};

set(gca,'yticklabel',s,'ytick',1:6,'ylim',[1 6]);

Figui
```



Example 23.3: Lunar-descent control simulation.

CHAPTER 24

James Webb Space Telescope ACS design

This chapter describes a design for the James Webb Space Telescope ACS. The great thing about NASA and ESA programs is that the engineers write numerous papers. These allow you to learn about spacecraft design. This chapter does not present the actual design used on the spacecraft, just a design that meets some of the published requirements. The design in this chapter is unique to this book. It also relies entirely on information available to the public. There are always multiple ways to design a spacecraft-control system.

The James Webb is an inertial precision-pointing spacecraft. It needs to point with extreme accuracy and with very low jitter at distant stars. The telescope is shown in Fig. 24.1.

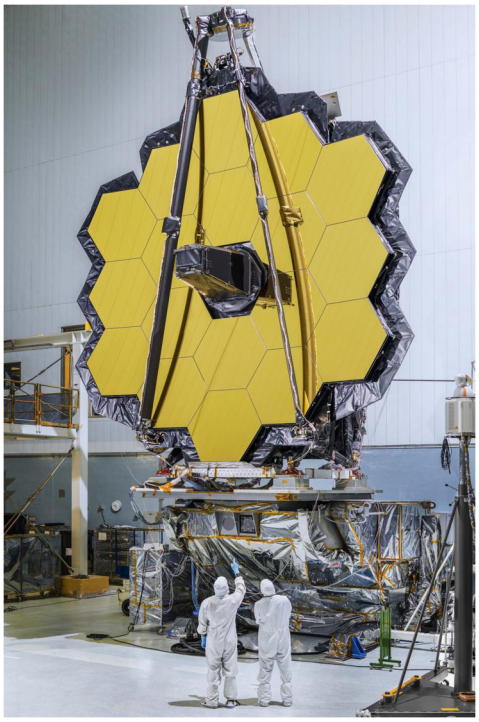


Figure 24.1 James Webb Space Telescope. Image courtesy of NASA.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00036-X

The structure of the control system is shown in Fig. 24.2. There are two control functions, momentum management and attitude control. Both produce torque-demand vectors. These are converted into thruster pulsewidth and reaction-wheel commands using linear programming. Sensing is done via star cameras and inertial measurement units (IMUs). The attitude determination system uses an unscented Kalman filter with the IMU as the dynamical base.

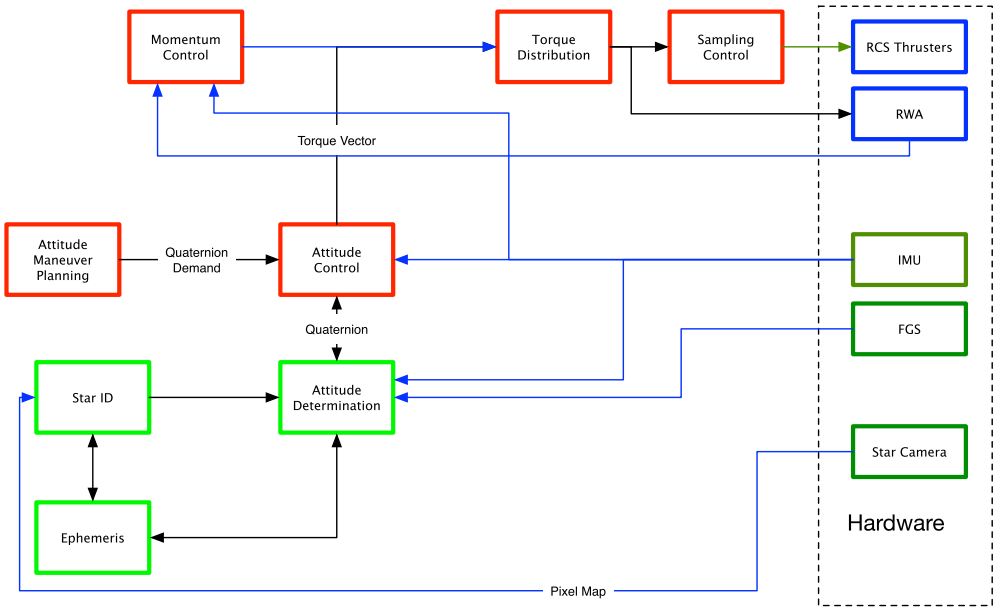


Figure 24.2 The James Webb Space Telescope ACS system designed in this chapter. FGS is a fine guidance sensor.

The design presented in this chapter uses the actuators and sensors selected for JWST. It does not use the same algorithms or control-system structure. Idealized models for all sensors and actuators are used.

24.1. Requirements

ACS requirements are summarized in Table 24.1 [1].

These requirements do not include jerk, the time derivative of acceleration, settling times, or jitter. Presumably, these were added in later iterations of the requirements.

Table 24.1 ACS requirements. FGS is a fine guidance sensor.

Requirement	Time
90 degree slew	60 minutes
Pointing	7 arcsec
Pointing FGS	1 arcsec
Settling time	
Jitter	
Jerk	

24.2. Spacecraft model

The James Webb CAD model is shown in Fig. 24.3. The telescope mirror is on the right. The large structure in the middle is the sunshade. The spacecraft bus, consisting of all the hardware needed to run the spacecraft, is on the left. The panel on the right bottom is the momentum-balance panel. James Webb uses the axes designations J1, J2, and J2. (+J1,+J2,+J3) are (+X,+Y,+Z) in the diagram. Fig. 24.3 shows the James Webb Coordinate system.

James Webb Space Telescope parameters are given in Table 24.2.

SCAT stands for Secondary Combustion Augmentation Thrusters. These use an oxidizer with hydrazine. This gives them a higher exhaust velocity, thus lowering fuel consumption. The bias rate ensures that the wheels do not pass through zero. This prolongs the life of the wheels and eliminates noise due to commutation issues at zero wheel speed. Most wheels use trapezoidal commutation that produces noisy wheel-speed measurements at low speed.

The thrusters are a blowdown system, shown in Fig. 24.4, so the thrust will decrease with time. Typically, blowdown systems operate between 350 and 100 psi.

24.3. Disturbances

The spacecraft resides at the L2 Lagrange point. The only disturbances that impact the attitude are

1. Solar torques;
2. Thermal torques;
3. RF torques.

Solar torques are from the interaction of the Sun with the surfaces. Thermal torques are due to emissions from radiators or other hot objects on the surface of the spacecraft. RF torques are due to RF emissions from antennas. Gravity gradient, albedo, planetary radiation, residual dipole, and drag torques do not exist at L2.

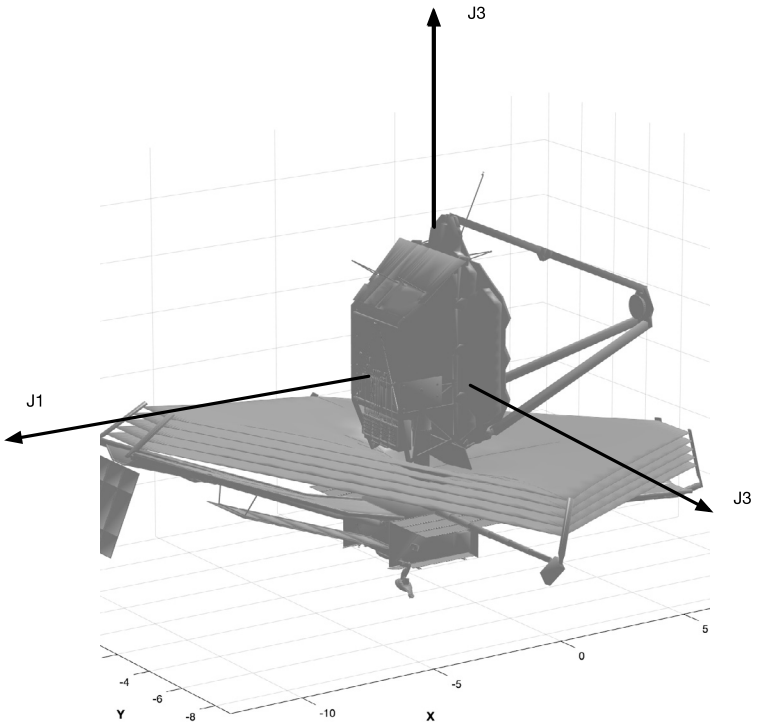


Figure 24.3 The James Webb Space Telescope CAD model. The surfaces are divided into triangles.

The thermal, solar, and RF torques are of the form

$$T = (r_k - r_c)^{\times} F_k \tag{24.1}$$

where r_c is the center-of-mass and r_k is the location of the element producing the force F_k. Solar forces are due to the Sun flux reflecting off the element. Fig. 24.5 shows the 2D layout.

The transmit power is quite low, 50 W, leading to a force of P/c (power divided by the speed-of-light) or 0.17 μN. The offset from the center-of-mass is about 3 m so the disturbance torque is 0.5 μN m. If we assume 9.5 hours of operation, the momentum build-up would be

$$h = tr\frac{P}{c} \tag{24.2}$$

where r is the moment arm and t is the time of operation. The total momentum growth is 0.017 N m s, which is negligible.

Table 24.2 James Webb Space Telescope parameters [2] relevant to ACS. Principal axes are found primarily through a rotation about the *y*-axis. The center-of-mass is for the CAD representation.

Parameter	Value	Units
Inertia	$\begin{bmatrix} 67\,946 & -83 & 11\,129 \\ -83 & 90\,061 & 103 \\ 11\,129 & 103 & 45\,821 \end{bmatrix}$	kg m^2
Inertia (Principal Axes)	$\begin{bmatrix} 72\,575 & 0 & 0 \\ 0 & 90\,061 & 0 \\ 0 & 0 & 41\,191 \end{bmatrix}$	kg m^2
Center-of-mass	$\begin{bmatrix} 0.46 \\ 0 \\ 2.58 \end{bmatrix}$	m
Momentum-wheel torque	0.072	N m
Wheel spin-axis Inertia	0.1295	kg m^2
Wheel bias rate	2700	rpm
Wheel maximum rate	6000	rpm
Number of wheels	6	
ACS thrusters (16)	4.45	N
Delta-V (SCAT) thrusters (4)	35	N
Dinitrogen tetroxide [3]	133	kg
Hydrazine [3]	168	kg
RF Power [4]	50	W
Data transmit time [4]	9.5	hours

JWST has very sophisticated radiators for its optical system [6], and heat-rejection systems for the spacecraft bus.

The solar-disturbance model can be quite simple since the Sun is always on one side of the sunshield. This means a single plate, defined by an area, surface properties, center of pressure, and location, can be used. The sunshield is shown in Fig. 24.6.

The sunshield [7] is 21.197 m × 14.162 m. The outermost layer has a "doped-silicon" coating. Assume that the reflection coefficients are purely specular. The solar force for pure specular reflection is

$$F_k = -2a_k \frac{p}{c}(s^T n_k)n_k \tag{24.3}$$

where s is the vector to the Sun, a_k is the area, and n_k is the normal. The model is shown in Fig. 24.7.

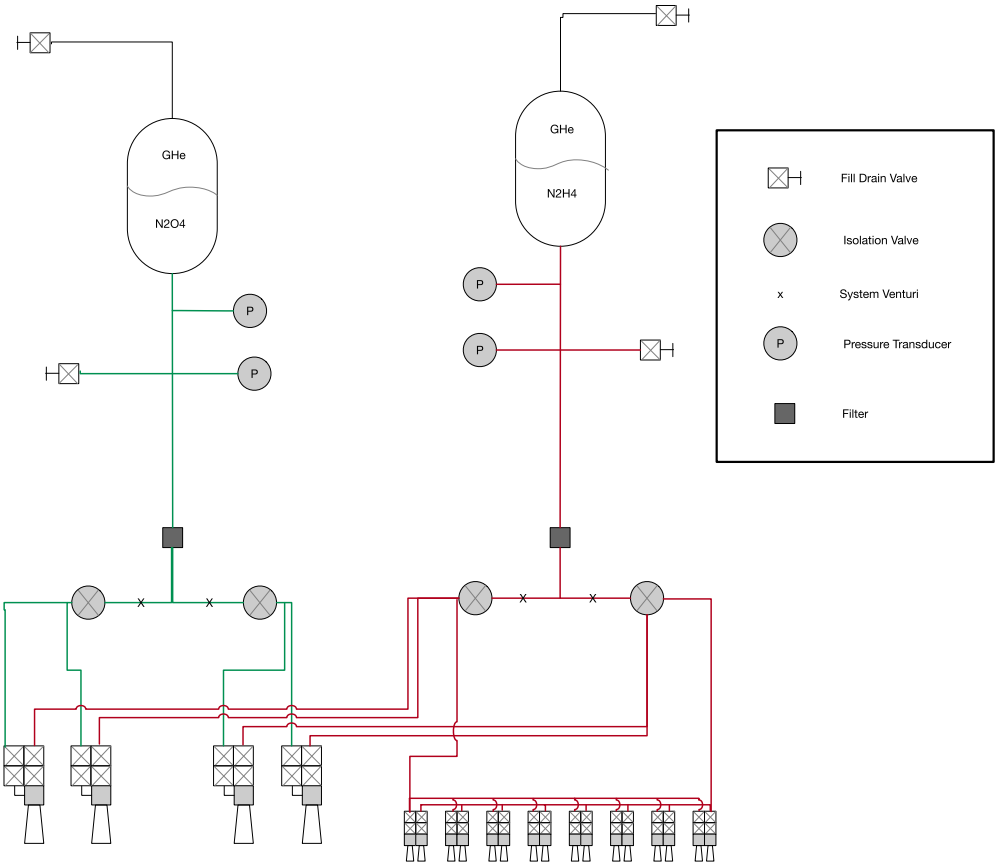


Figure 24.4 Propulsion system [5].

Example 24.1 gives a disturbance analysis. The center-of-mass is chosen so that the flap exactly balances the solar-pressure torques when the Sun is along the $-z$-axis. The script shows the variation for slight y angles.

24.4. Attitude maneuvers

The attitude-maneuver requirement is a 90-degree slew in 60 minutes. Considerable work has been done on slewing JWST [8,2]. For this analysis, we will assume there are no requirements on jerk, the time derivative of angular acceleration, so that we can use a bang-bang maneuver that is filtered to not excite slosh dynamics. For constant acceleration, the required torque is

$$T = \frac{4I\theta}{t^2} \tag{24.4}$$

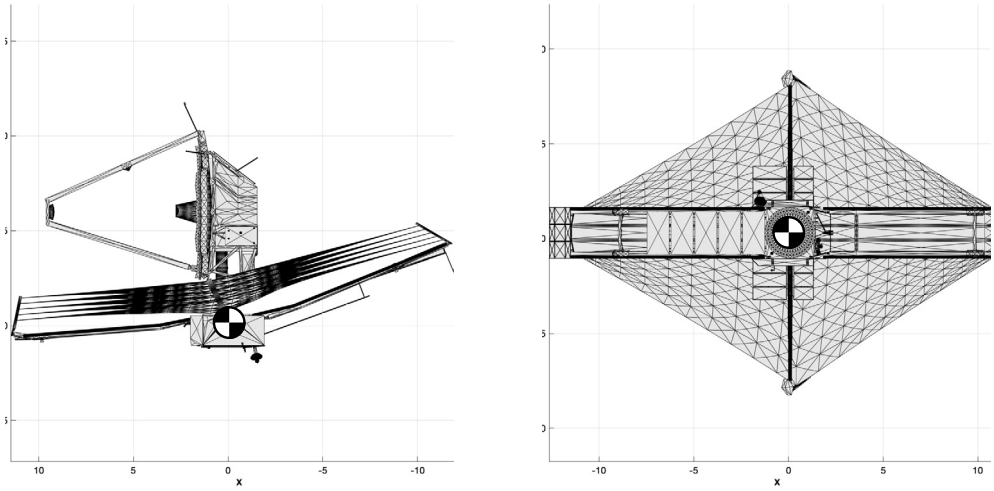


Figure 24.5 James Webb Space Telescope 2D drawing showing the approximate center-of-mass.

Figure 24.6 The James Webb Space Telescope sunshield. Image courtesy of NASA.

where t is the maneuver duration, I is the inertia, and θ is the maneuver angle. The maximum momentum stored for the maneuver is

$$h = \frac{Tt}{2} \tag{24.5}$$

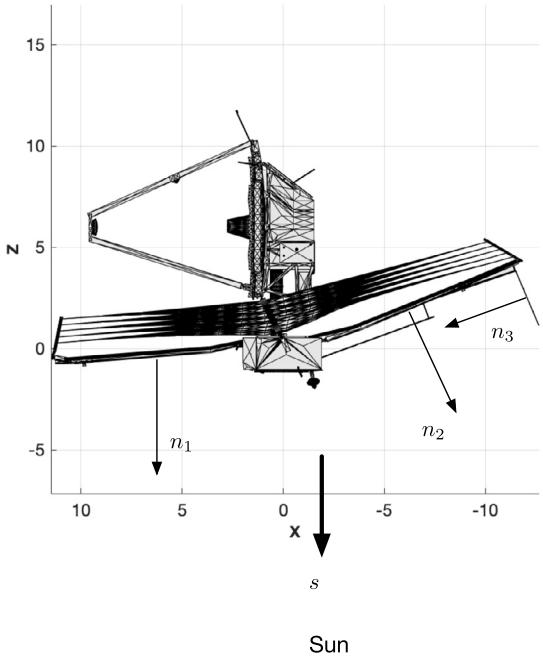


Figure 24.7 Disturbance model with three plates.

The largest inertia is I_{yy} in the principal axes, which is 90 061 kg m^2. The torque is 0.044 N m and the required momentum storage is 78 N m s. The maximum momentum that can be stored in a wheel is 44.7 N m s. This means at least two wheels are required for a maneuver of this size.

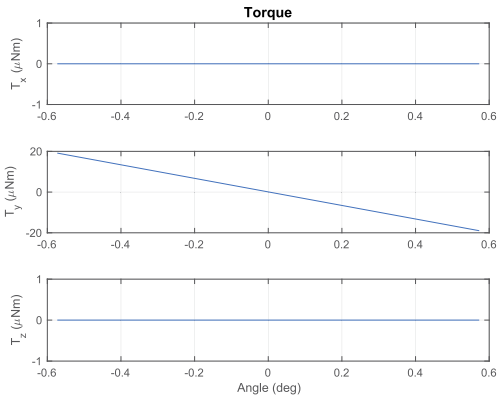
24.5. Momentum control

Momentum control is done with thrusters and the momentum panel. Normally thruster unloading would be done with the 4.5-N thrusters. However, momentum can also be unloaded during station-keeping maneuvers using a combination of SCAT and ACS thrusters.

24.6. Attitude control

Reaction wheels are used for normal pointing control, including all attitude maneuvers. Thrusters are used for momentum unloading and also for attitude control during station-keeping maneuvers. Using the reaction wheels does not make much sense since the momentum gained during the maneuver would need to be unloaded with thrusters later.

```
%% Disturbance analysis

shade2Ang   = 20*pi/180;% rad
flapAng     = 25*pi/180; % rad
yFlap       = 1.66+1.04;
lFlap       = sqrt((12.49-11.46)^2 + (3.82-1.48)
    ^2);
aFlap       = yFlap*lFlap;
aShade      = 21.2*14.2;
cM          = [-0.5848;0;2.58];
p           = 1367/3e8;
s           = [0;0;-1];
n           = [[0;0;-1] [-sin(shade2Ang);0;-cos(
    shade2Ang)] [cos(flapAng);0;-sin(flapAng)
    ]];
a           = [aShade/2 aShade/2 aFlap];
r           = [[ 5;0;0.66] [-5;0;1.35]
    [-12;0;2.47]];

% The CM with the sunshade
cM(1)       = fminsearch(@fun,-0.5,[],n,a,r,p,s,cM
    );

% Shade 1, Shade 2, Flap
ang         = [-0.01 0 0.01];
torque      = zeros(3,length(ang));
for j = 1:length(ang)
  s           = [sin(ang(j));0;-cos(ang(j))];
        torque(:,j) = TorqueCalc(n,a,r,p,s,cM);
end

Plot2D(ang*180/pi,torque*1e6,'Angle (deg)',...
  {'T_x (\muNm)' 'T_y (\muNm)' 'T_z (\muNm)'},'
        Torque');

function y = fun(x,n,a,r,p,s,cM)

cM(1)  = x;
torque = TorqueCalc(n,a,r,p,s,cM);
y      = abs(torque(2));

end

function torque = TorqueCalc(n,a,r,p,s,cM)

torque =[0;0;0];
for k = 1:3
        f      = -2*p*a(k)*(n(:,k)'*s)*n(:,k);
        torque = torque + Cross(r(:,k)-cM,f);
end

end
```



Example 24.1: Disturbance torque as a function of Sun angle.

24.7. Torque distribution

Torque distribution uses linear programming. Reaction wheels produce variable torques based on the voltage applied to the motor. The thrusters are on/off, so pulsewidth modulation is used to generate "proportional" torques. Torque distribution uses linear programming. This is

$$x = Au \tag{24.6}$$

$$\min(c^T u) \tag{24.7}$$

$$u \geq 0 \tag{24.8}$$

The A matrix is a 3-by n array of torque vectors giving the maximum vector of each torque. For the wheels, we add a positive and negative torquer-vector set. This allows the algorithm to produce negative reaction wheel commands. We put the six wheels at 45-degree angles with respect to the z-axis and oriented them in a hexagon around z. The thrusters are placed on the four sides of the bus. They are angled so that their plumes go in the -z-direction. Fig. 24.8 shows the layout.

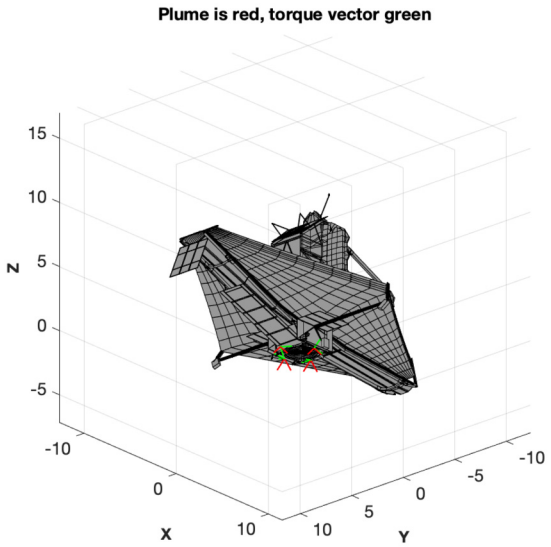


Figure 24.8 Thruster layout. There are eight thruster pairs in four locations at the bottom edge of the bus. The thrust vectors are all at 45 degrees with respect to the sunshade.

24.8. Attitude determination

Fig. 24.9 shows the star-camera-based attitude determination system.

It is useful to study the Kalman filter using a single-axis filter. The IMU model is

$$\dot{\theta}_i = \omega + b + \eta_{\theta_i} \tag{24.9}$$

$$\dot{b} = \eta_b \tag{24.10}$$

where ω is the true body rate. The IMU outputs an integrated rate. b is the bias, which is driven by a random-walk process. η_{θ_i} and η_b are Gaussian white noise. The IMU model is linear and the measurement is assumed to be the angle θ_b.

$$\gamma = \theta_b + \eta_{\theta_m} \tag{24.11}$$

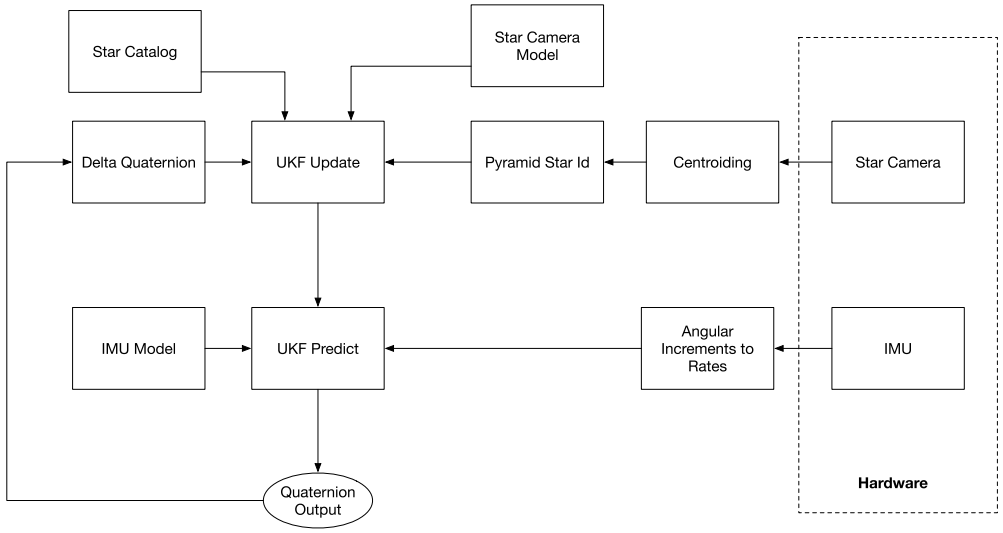


Figure 24.9 Attitude determination using an unscented Kalman filter (UKF).

where θ_b is the actual angle and η_{θ_m} the measurement noise. The model is in the form

$$\dot{x} = ax + bu \tag{24.12}$$

$$y = cx \tag{24.13}$$

$$a = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{24.14}$$

$$b = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{24.15}$$

$$c = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{24.16}$$

$$x = \begin{bmatrix} \theta \\ b \end{bmatrix} \tag{24.17}$$

This single-axis model assumes that the star-camera measurement produces an angle as the product.

The model needs to be converted to discrete time for a standard Kalman filter. A standard Kalman filter will work for this single-axis, system. Note that

$$\dot{\theta}_b = \omega_b \tag{24.18}$$

The Kalman filter has two parts, state prediction and measurement incorporation. The Kalman-filter measurement incorporation is

$$k = \frac{ph^T}{hph^T + r} \tag{24.19}$$

$$x_e = k(y - hx_e) \tag{24.20}$$

$$p = (E - kh)p \tag{24.21}$$

where E is a 2-by-2 identity matrix, p is the covariance matrix, r is the measurement covariance matrix, and the measurement matrix h is

$$h = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{24.22}$$

which is the same as the c matrix. The measurement is just θ_b. The prediction step is

$$x_e[k] = ax_e[k] + b\left(\frac{\theta_i[k] - \theta_i[k-1]}{\Delta t}\right) \tag{24.23}$$

$$p = apa^T + q \tag{24.24}$$

where a is the state-transition matrix, q is the model covariance matrix, and Δt is the time step. The output of the rate-integrating gyro is differenced to get the rate. This is where the IMU connects to the model. The IMU output is not a measurement.

Example 24.2 implements the Kalman filter. The script gives three choices for IMUs. The superior noise-filtering characteristics of the hemispherical resonating gyro are evident.

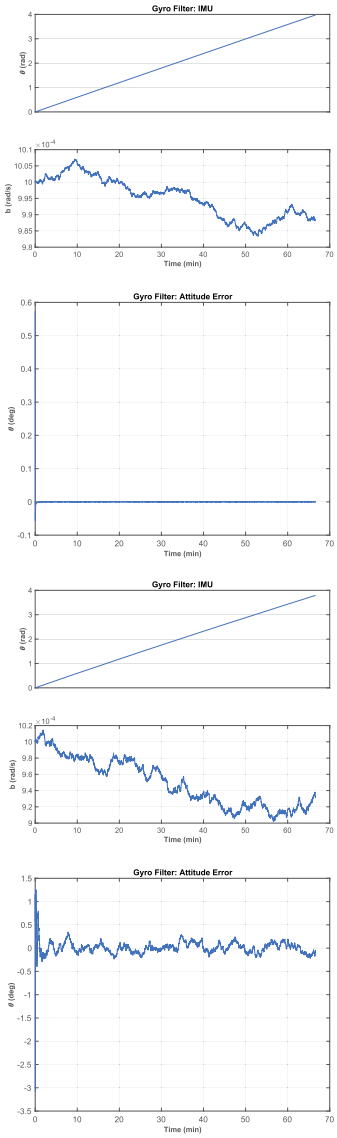
24.9. Simulation

This section shows a maneuver with a failure of a reaction wheel during a reorientation.

For James Webb, the major physical effects, beyond the rigid body are reaction wheels, slosh, and flexibility. The spacecraft has two fuel tanks, one for hydrazine and the other for the SCAT oxidizer. The spacecraft has six reaction wheels.

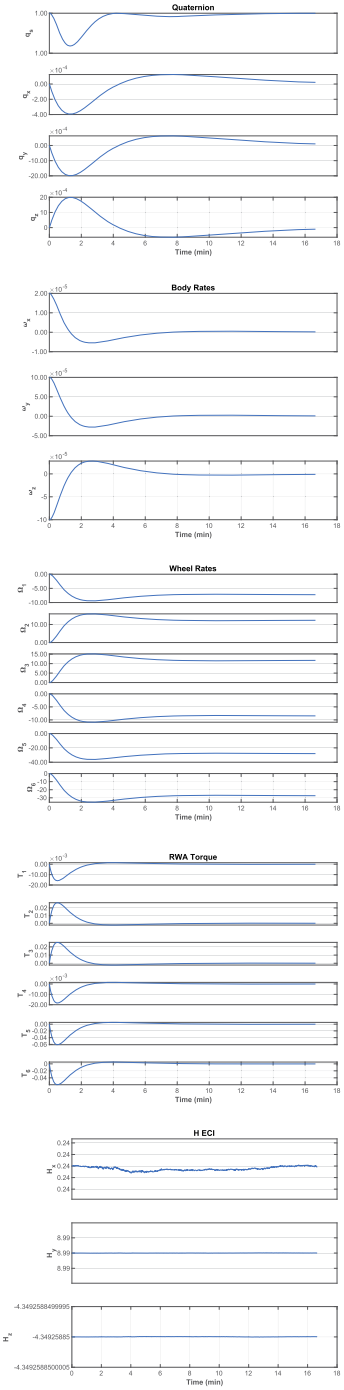
The simulation does not include slosh or flexibility. Six reaction wheels are employed to control the spacecraft using a PID controller. A momentum controller using thrusters is employed. The response is what would be expected from a PID controller. The pseudoinverse distributes the torque among the six wheels that are oriented in a pyramid. Fewer wheels could be used but they might not have sufficient momentum storage or torque capabilities. Six wheels also provide redundancy. The last plot shows the inertial angular momentum. Since all torques are internal, it is conserved.

```
model        = 'MEMS'; % 'adi' 'HRG', 'MEMS'
n            = 4000; % Number of time steps

%% IMU Model
h            = [1 0]; % Measurement matrix
a            = [0 1;0 0];
b            = [1;0];
dT           = 1;
[phi,gamma] = C2DZOH(a,b,dT);

%%
p            = 0.001*eye(2); % Set the initial
    covariance
switch(lower(model))
  case 'adi'
    d = struct('etaB',RW2SDev(2),'etaTheta',0.75*
        pi/180,'u',0,'omega',0);
    sigmaY = pi/180;
  case 'hrg'
    d = struct('etaB',RW2SDev(0.001),'etaTheta'
        ,60*0.02*4.85E-6,'u',0,'omega',0);
    sigmaY = 4.85E-6; % 1 arc-second in radians
  case 'mems' % CRS39A Series Silicon Sensing
    d = struct('etaB',RW2SDev(0.006),'etaTheta'
        ,0.006*pi/180,'u',0,'omega',0);
    sigmaY = pi/180; % 1 arc-second in radians
end

r = sigmaY^2;
q = diag([d.etaTheta d.etaB].^2); % Plant noise
    covariance

%% Simlation
x = [0;0.001];
xE = [0.01;0.005];
t = (0:n-1)*dT;

% For rate estimation from the angle
x1Old = 0;

% Run the filter for n steps
xP = zeros(7,n);
for j = 1:n

  % True angle
  theta    = d.omega*t(j);

  % Measurement
  y        = theta + sigmaY*randn;

  % Plot storage
  xP(:,j) = [x;xE;y;diag(p)];

  % Measurement incorporation
  k        = p*h'/(h*p*h' + r);
  xE       = xE + k*(y - h*xE);

  % Measurement update of covariance
  p        = (eye(2) - k*h)*p;

  % State propagation Input is the gyro angle
  xE       = phi*xE + gamma*(x1Old-x(1))/dT;
  p        = phi*p*phi' + q;
  x1Old    = x(1);

  % Integrate
        x        = RK4(@RIG,x,dT,0,d);

end

theta = d.omega*t;
delta = (xP(3,:)-theta)*180/pi;

leg = {'True' 'Estimated'};
leg = {leg leg};
yL = {'\theta (rad)','b (rad/s)',...
  '\theta_E (rad)','b_E (rad/s)',...
  'y (rad)' 'p_\theta','p_b'};
TimeHistory(t,xP(1:2,:),yL(1:2),'IMU');
TimeHistory(t,xP(6:7,:),yL(6:7),'Covariance');
TimeHistory(t,delta,{'\theta (deg)'},'Error');

%% Gyro RHS
function xDot = RIG( x, ~, d )

b          = x(2);
bDot       = d.etaB*randn;
thetaDot   = d.omega + b + d.etaTheta*randn + d.u;
xDot       = [thetaDot;bDot];

end
```



Example 24.2: Single-axis Kalman filter. The first two plots are for the HRG. The second two for a good-quality MEMS IMU.

```
%% Constants
secInDay   = 86400;
controlOn  = true;
mmOn       = true; % Momentum management

%% User initialization
n          = 1000; % Number of simulation steps
dT         = 1.0; % Time step (s)
inertia    = [67946 -83 11129;-83 90061 103;11129
       103 45821];
qTarget    = [1;0;0;0];

% Initial Julian date
jD0        = Date2JD([2023 4 4]);

% Initial states
q          = [1;0;0;0];
omega      = [0;0;0]; % rad/s
dOmega     = [0.00002;0.0001;-0.0001]; % Delta
       angular rate
omegaRWA   = [0;0;0;0;0;0]; % rad/s
torqueD    = [2;0;0]*1e-6; % Disturbance torque

%% State vector
x      = [q;omega+dOmega;omegaRWA];
dRHS = RHSJWST;
dRHS.inertia = inertia;

%% Simulate
xP         = zeros(25,n);

% Edit the control system data structure
dC         = ACSJWST;
dC.dPID.q_desired_state = qTarget;

dC.dT      = dT;
dC.uRWA    = dRHS.uRWA;
dC.mmOn    = mmOn;
dC.dPID.inertia   = dRHS.inertia;

% Simulation loop
for k = 1:n
  if( controlOn )
    [dC,tThruster,tRWA] = ACSJWST(x,dC);
  else
    tThruster = [0;0;0]; tRWA = [0;0;0;0;0;0];
  end
  dRHS.torque       = tThruster + torqueD;
  dRHS.tRWA         = tRWA;
  [~,hECI]          = RHSJWST(x,0,dRHS);
  xP(:,k)           = [x;tThruster;tRWA;hECI];
  x                 = RK4(@RHSJWST,x,dT,0,dRHS);
end

%% Plotting
t  = (0:n-1)*dT;

yL = {'q_s' 'q_x' 'q_y' 'q_z' ...
'\omega_x' '\omega_y' '\omega_z' ...
'\Omega_1' '\Omega_2' '\Omega_3' ...
'\Omega_4' '\Omega_5' '\Omega_6' ...
'T_x' 'T_y' 'T_z' ...
'T_1' 'T_2' 'T_3' 'T_4' 'T_5'  'T_6'...
'H_x' 'H_y' 'H_z' };

tL = {'Quaternion' 'Body Rates' 'Wheel Rates' ...
      'REA Torque' 'RWA Torque' 'H ECI' };

kL = {1:4 5:7 8:13 14:16 17:22 23:25};
for j = 1:length(tL)
  k = kL{j};
  TimeHistory(t,xP(k,:),yL(k),tL{j});
end

Figui
```



Example 24.3: JWST simulation. The PID controller damps rates using the six reaction wheels.

References

[1] B. Bogenberger, James Webb Space Telescope Project Mission Requirements Document, Tech. Rep. JWST-RQMT-000634 Revision P, Goddard Space Flight Center, October 2007.
[2] M. Karpenko, J.T. King, C.J. Dennehy, I. Michael Ross, Agility analysis of the James Webb Space Telescope, Journal of Guidance, Control, and Dynamics 42 (4) (April 2019) 810–821.
[3] Anonymous, Webb fuelled for launch, Tech. Rep., European Space Agency, June 2021.
[4] J. Gal-Edd, E. Luers, James Webb Space Telescope Ka-Band Trade, Tech. Rep. 20040035670, NASA Goddard Spaceflight Center, January 2004.
[5] J.D. Petersen, J. Tichy, G.G. Wawrzyniak, K. Richon, James Webb Space Telescope initial mid-course correction Monte Carlo implementation using task parallelism, in: International Symposium on Space Flight Dynamics, 2014.
[6] R.A. Franck, A.P. Gurule, P.A. Brinckerhoff, T.R. McCallan, M. Renbarger, A.A. Brown, High performance cryogenic radiators for James Webb Space Telescope, in: 46th International Conference on Environmental Systems, No. ICES-2016-141 in 1, July 2016.
[7] GSFC, About the sunshield, https://jwst.nasa.gov/content/observatory/sunshield.html.
[8] K.Q. Ha, M.D. Femiano, G.E. Mosier, Minimum-time and vibration-avoidance attitude maneuver for spacecraft with torque and momentum limit constraints in redundant reaction wheel configuration, in: L.D. Peterson, R.C. Guyer (Eds.), Space Systems Engineering and Optical Alignment Mechanisms, in: Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 5528, Oct. 2004, pp. 126–137.

CHAPTER 25

CubeSat control system

25.1. Space story

In 2010 Princeton Satellite Systems established a CubeSat club for middle-school students at the John Witherspoon Middle School in Princeton, New Jersey, USA. It was an after-school club open to all students. We ran the club in the school shop classroom. The club ran from 2010 through 2012. The students divided themselves into subsystem groups. The students built CubeSat hardware and wrote CubeSat software in MATLAB® and C. The shop teacher and the science teacher, Steve Carson, were the faculty members. Eloisa de Castro, a Princeton Satellite Systems engineer, gave a talk about the club at a National Science Foundation meeting.

Fig. 25.1 shows magnetic torquers built by the students. The winding rig was built by the shop teacher. Fig. 25.2 shows a solar-cell simulator and reaction-wheel hardware connected to a simulation.

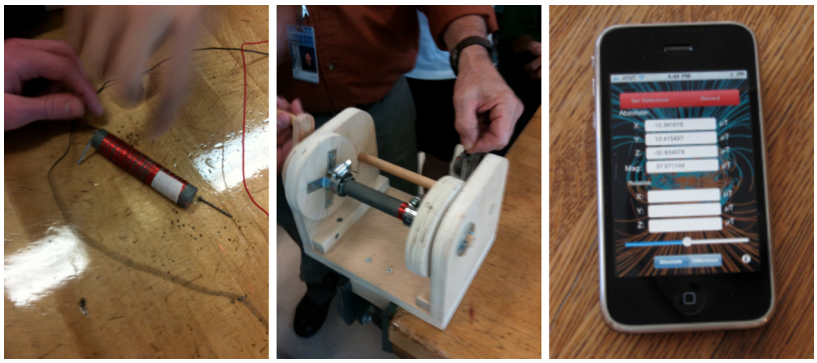


Figure 25.1 Student-built magnetic torquers and field-measurement app on an iPhone.

The club had a visit from Bob Cenker, who orbited Earth on the Space Shuttle Columbia. The students had the chance to ask Bob plenty of questions about his experience as an astronaut and they even had a discussion about orbital mechanics.

The club was a big success. Most students stayed for all four years. After the club ended, parents called us to ask if we could restart it. Today, there are many CubeSat activities for elementary, middle, and high-school students.

https://doi.org/10.1016/B978-0-32-399915-1.00037-1

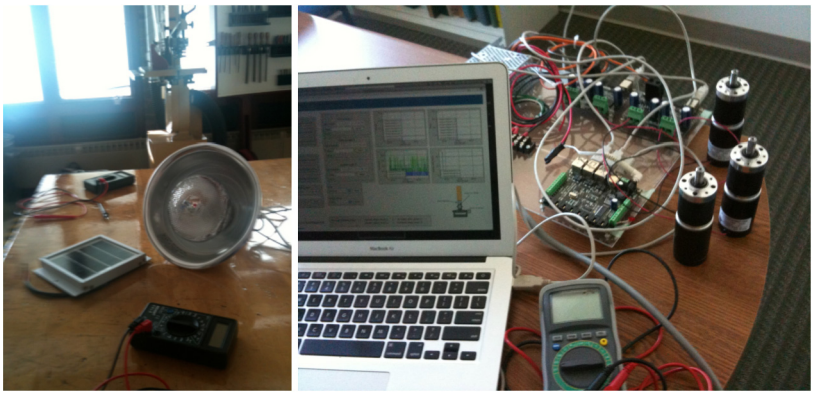


Figure 25.2 A solar cell and simulator and reaction-wheel hardware connected to a simulation.

25.2. Introduction

CubeSats are used for a wide variety of applications. The control systems range from simple gravity-gradient systems to elaborate three-axis stabilized designs with reaction wheels and star trackers. Many companies supply low-cost hardware for CubeSat applications. CubeSats are often used in LEO applications, but have been used for cutting-edge missions, like recent CubeSats that were part of a Mars mission.

This chapter describes a CubeSat control-system design. The design uses three reaction wheels for attitude control. An Earth sensor and magnetometer are used for sensing. Gyros are used as part of a Kalman filter to filter the sensor measurements. Magnetic torquers are used to unload momentum from the reaction wheels.

25.3. Requirements

The CubeSat is Earth pointing. The control requirements are to acquire the Earth and then maintain pointing.

25.4. Actuator and sensor selection

There are four attitude control devices

1. Magnetic torquers;
2. Fisheye Earth sensor;
3. Magnetometer;
4. Reaction wheels.

The concept of operations is shown in Fig. 25.3. This gives the sequence of events from separation from the launch vehicle to the operational configuration.

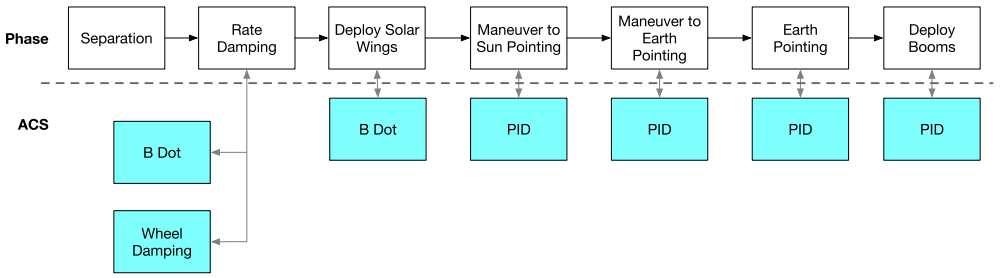


Figure 25.3 CubeSat concept of operations. Damping of rates is done at separation. The spacecraft deploys its solar wings, then reorients.

The CubeSat separates in an arbitrary orientation with arbitrary body rates. The spacecraft has two damping choices. If the body rates are low enough, it can absorb the momentum in the wheels and then proceed with operations. That would be the fastest approach. If the momentum is too high, it applies the B Dot algorithm, found in Section 12.11.1, using the magnetic torquers. The advantage of B Dot is that only the magnetometer and the magnetic torquers are needed. This has one major advantage over using the gyros because the gyro rates may have a bias, making it difficult to fully damp the body rates. Without external angle measurements, it is not possible to estimate the bias or correct for bias random walk.

25.5. Design

The CubeSat is shown in Fig. 25.4. The solar wings are 30 cm by 10 cm. They are body fixed.

25.6. Control-system design

The control system is shown in Fig. 25.5. It has a magnetometer, Earth sensor, reaction wheels, and magnetic torquers. The reaction wheels provide 3-axis control. The magnetic torquers provide momentum unloading. They can also be used for backup attitude control.

The control system uses a proportional-derivative (PD) controller. This will not completely cancel constant torque offsets. A constant disturbance torque will result in a constant attitude error.

$$\frac{T}{\theta_y} = K\left(1 + \frac{\tau_r s}{\tau_f s + 1}\right) \tag{25.1}$$

where T is the control torque, θ_y the pitch angle, τ_r the rate time constant, K is the forward gain, τ_f the rate filter time constant, and s is the derivative, i.e., $\frac{d}{dt}$. The rate filter

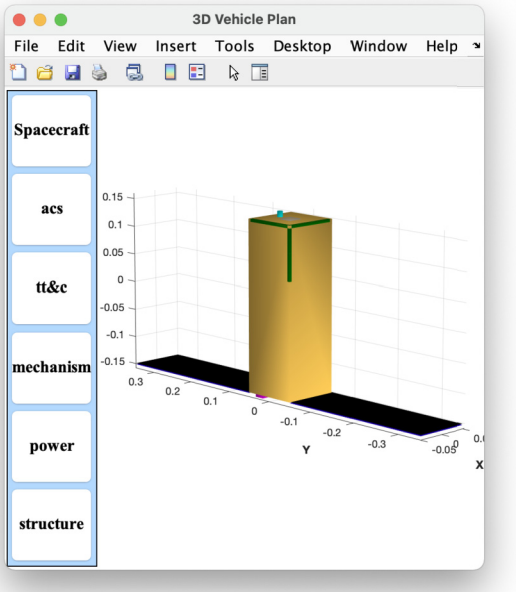


Figure 25.4 3U CubeSat with deployable solar wings. The wings are fixed. The orthogonal torquers and Earth sensor are on the nadir deck. The magnetometer is on the zenith deck.

allows the derivative to be implemented in state-space form. It also provides filtering as the differentiator amplifies noise. The wheels can also be used just for damping, in which case the control algorithm is

$$\frac{T}{\theta_y} = K\left(\frac{\tau_r s}{\tau_f s + 1}\right) \tag{25.2}$$

Attitude determination is done using an Earth sensor and magnetometer. This provides 3-axis information throughout the orbit. It requires knowing the orbital position, for the magnetic-field model, so a GPS receiver, or a reasonably accurate onboard ephemeris, is required. The momentum-control system uses a PI controller. The integral term removes any momentum offset due to steady inertial torques.

$$\frac{T}{H} = K\left(1 + \frac{1}{\tau_i s}\right) \tag{25.3}$$

where H is the inertial angular momentum. The closed-loop momentum equation is

$$\left(s + K\left(1 + \frac{1}{\tau_i s}\right)\right) H = T_d \tag{25.4}$$

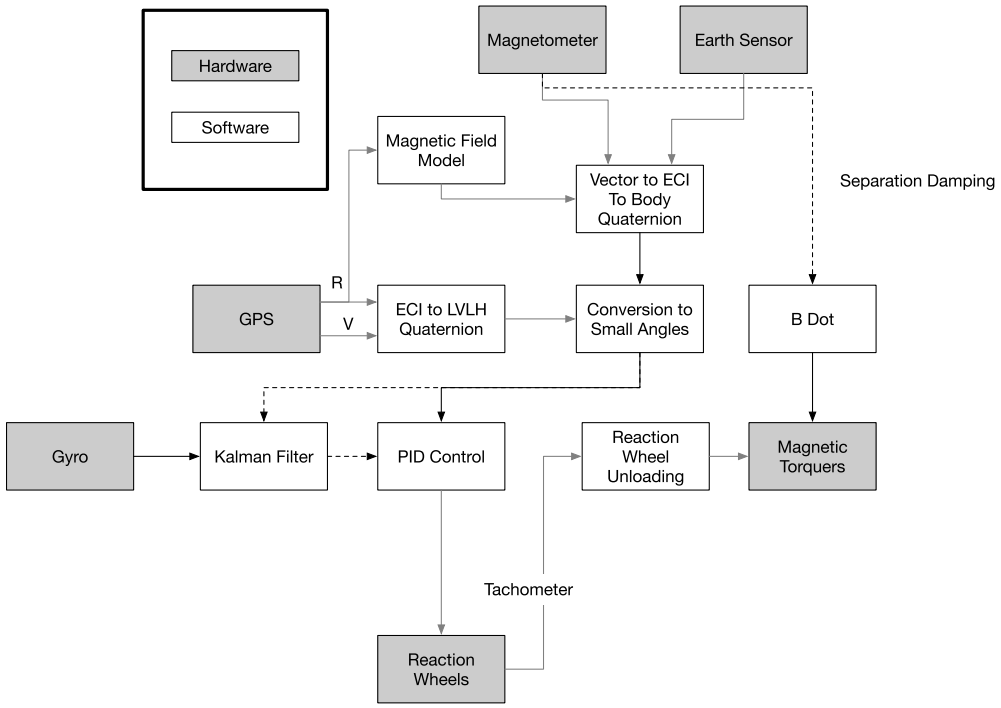


Figure 25.5 The CubeSat control system. It has a magnetometer, Earth sensor, reaction wheels, and magnetic torquers.

without the integral term, the offset is

$$H = \frac{T_d}{K} \tag{25.5}$$

With the integral term

$$\left(s^2 + K\left(s + \frac{1}{\tau_i}\right)\right) H = sT_d \tag{25.6}$$

the response is

$$H = \frac{sT_d}{s^2 + Ks + \frac{1}{\tau_i}} \tag{25.7}$$

The derivative in the numerator means that there will be no momentum offset due to a constant disturbance torque, T_d.

25.7. Attitude determination

The attitude determination system uses a single frame of data to get the quaternion from ECI to body. The CubeSat magnetometer always produces a vector measurement, but the Earth sensor only works when the Earth is in the field-of-view. The attitude determination can return either the ECI to body quaternion or ECI to LVLH quaternion. For this reason, one Earth sensor is mounted on the zenith face and one on the nadir face. This insures that a nadir vector can be measured at all times.

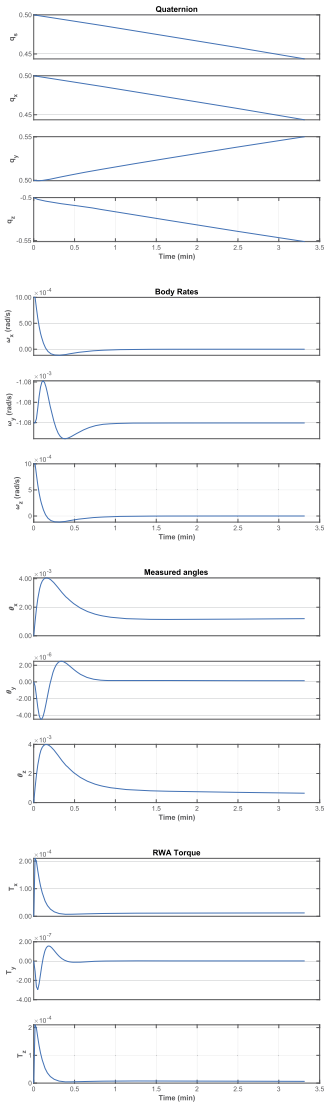
25.8. Simulation

Example 25.1 shows the attitude control system simulation. The quaternion from ECI to body changes throughout the simulation because the spacecraft, which is aligned with the LVLH frame, is always rotating with respect to the ECI frame. The body rate is nominally constant about pitch.

The control system employs direct angle measurements in the LVLH frame.

The script in Example 25.1 allows the initial angular rate to be easily changed and for a disturbance torque to be added. These allow the disturbance response of the control system to be tested. The plots in this example show the transient response from an initial rate error and the rate-damping performance of the PD controller.

```
secInDay   = 86400;
controlOn  = true;
mmOn       = false; % Momentum management
n          = 200; % Number of simulation steps
dT         = 1.0; % Time step (s)
omegaC     = 0.1; % Control bandwidth
zetaC      = 1; % Control damping ratio
jD0        = Date2JD([2023 4 4]);

% Initial states
el         = [7000 0 0 0 0 0]; % LEO
[r,v]      = El2RV(el);
orbRate    = 2*pi/Period(el(1));
omega      = [0;-orbRate;0]; % rad/s
dOmega     = [0.001;0;0.001]; % Delta angular rate
omegaRWA   = [0;0;0]; % rad/s
torqueD    = [0;0;0]*1e-6; % Disturbance torque
eSANoise   = [1e-5;1e-5]*1e-10; % ESA noise
bNoise     = 1.e-12; % Magnetometer noise
b          = [0;0;0]; % IMU bias at start
imuAngle   = [0;0;0]; % IMU angle at start
q          = QLVLH(r,v);
x          = [r;v;q;omega+dOmega;omegaRWA;b;
              imuAngle];
dRHS       = RHSReactionWheelWithOrbit;

%% Simulate
xP         = zeros(length(x)+15,n);
% Edit the control system data structure
dC         = ACSCubeSat;
dC.inr     = dRHS.inr;
dC.dT      = dT;
dC.omega   = omega; % The LVLH rate
dC.uRWA    = dRHS.uRWA;
dC.mmOn    = mmOn;
dC.kF      = dRHS.inr(2,2)*omegaC^2;
dC.tauR    = 2*omegaC*zetaC*dRHS.inr(2,2)/dC.kF;

% Create a time vector
t  = (0:n-1)*dT;
jD = jD0 + t/secInDay;

% Simulation loop
for k = 1:n
  r            = x(1:3);
  v            = x(4:6);
  qECIToBody   = x(7:10);
  % Sensor measurements
  bMeas        = QForm(qECIToBody,BDipole(r,jD(k))) +
                 bNoise*randn(3,1);
  dC.angle     = AngleDetermination(r,v,Unit(-QForm(
                 qECIToBody,r)),bMeas,jD(k));
  % Update the controller
  dC.qECIToBody = qECIToBody;
  if( controlOn )
    [dC,tThruster,tRWA] = ACSCubeSat(x,dC);
  else
    tThruster = [0;0;0]; tRWA = [0;0;0];
  end
  dRHS.torque = tThruster + torqueD; % Nm
  dRHS.torqueRWA = tRWA;
  [~,hECI] = RHSReactionWheelWithOrbit(x,0,dRHS);
  xP(:,k) = [x;bMeas*1e9;dC.angle;tThruster;tRWA;
             hECI];
  x = RK4(@RHSReactionWheelWithOrbit,x,dT,0,dRHS)
      ;
end

%% Plotting
yL = {'x_□(km)'  'x_□(km)' 'x_□(km)',...
'v_x_□(km/s)' 'v_y_□(km/s)' 'v_z_□(km/s)', ...
'q_s' 'q_x' 'q_y' 'q_z' ...
'\omega_x_□(rad/s)' '\omega_y_□(rad/s)'...
'\omega_z_□(rad/s)' ...
'\Omega_1' '\Omega_2' '\Omega_3' ...
'\theta_{ix}' '\theta_{iy}' '\theta_{iz}'...
'bias_{ix}' 'bias_{iy}' 'bias_{iz}'...
'b_x_□(nT)'  'b_y_□(nT)'  'b_z_□(nT)' ...
'\theta_x', '\theta_y', '\theta_z',...
'T_x' 'T_y' 'T_z' 'T_x' 'T_y' 'T_z'...
'H_x' 'H_y' 'H_z' };

tL = {'Position' 'Velocity' 'Quaternion'...
'Body□Rates' 'Wheel□Rates' 'IMU'...
'Magnetic□Field□Body□Frame' 'Measured□angles'...
'REA□Torque' 'RWA□Torque' 'H□ECI' };
kL = {1:3 4:6 7:10 11:13 14:16 17:22 23:25 26:28
      29:31 32:34 35:37};
for j = 1:length(tL)
  k = kL{j};
  TimeHistory(t,xP(k,:),yL(k),tL{j});
end

Figui
```



Example 25.1: Attitude control simulation.

CHAPTER 26

Microwave Anisotropy Satellite

This chapter describes the design of the Wilkinson Microwave Anisotropy (WMAP) Satellite ACS. The system discussed in this chapter is not the ACS that was used on the actual spacecraft. It shows the design process for designing a system that meets the requirements. The actual ACS is discussed in this paper [1]. The attitude determination system is also not discussed in this chapter. It is assumed that the attitude determination system produces an accurate quaternion from the ECI frame to the body frame.

26.1. The WMAP mission

The WMAP mission was designed to make fundamental cosmological measurements of the cosmic background radiation. As such, it needs to be able to point in an arbitrary inertial direction. WMAP produces a map of the cosmic microwave background radiation over the celestial sphere through a fast spin about its spin-axis and a slow precession of its spin axis about the sunline.

26.2. ACS overview

For this example, we replace the reaction wheels with momentum wheels.

The Wilkinson Microwave Anisotropy Probe is a three-axis controlled spacecraft. Fig. 26.1 shows the spacecraft.

The spacecraft attitude control system has

1. Two inertial measurement units (IMUs);
2. Two star trackers;
3. One digital Sun sensor;
4. Twelve coarse Sun sensors;
5. Three reaction-wheel assemblies;
6. A hydrazine monopropellant propulsion system.

The star trackers and IMUs provide the inertial reference. The Sun sensors provide the Sun vector. The reaction wheels are used for control and the thrusters for unloading.

26.3. Control modes

The WMAP spacecraft requires three control modes:

- Mission control mode using reaction wheels with thrusters for momentum unloading. This mode would also include the acquisition function.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00038-3

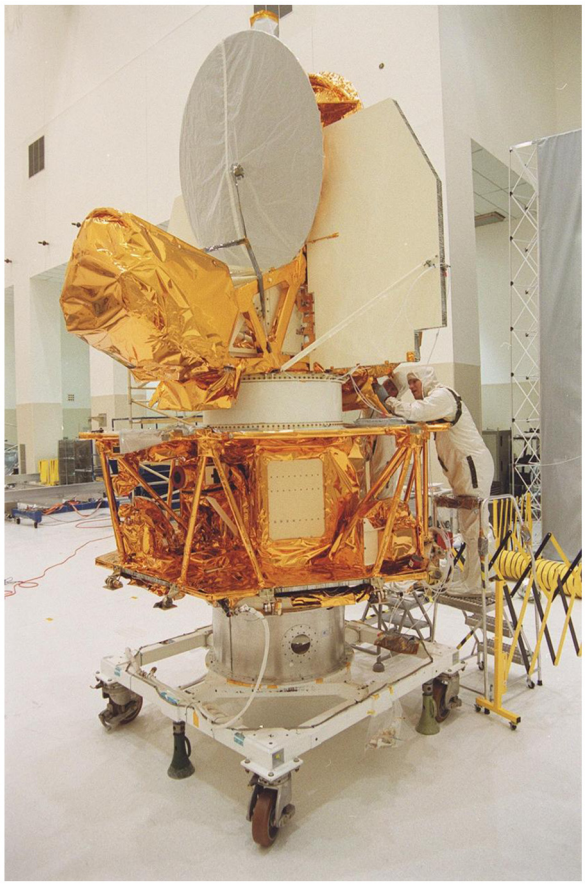


Figure 26.1 The Microwave Anisotropy Probe. Image courtesy of NASA.

- Backup mission control mode with thrusters for three-axis attitude.
- Orbit-adjust mode using thrusters for three-axis attitude control. The reaction wheels are kept in a tachometer mode during orbit changes.

Optionally, a safe-hold mode could be added. The safe mode would be to point the solar panels at the Sun and rotate about the sunline.

26.4. Sensing and actuation

Nominally, WMAP uses gyros for attitude determination and a star tracker to correct for gyro drift and to obtain absolute attitude information. The gyro and star-tracker data could be integrated using an extended iterated Kalman filter. A backup mode using just the star tracker would also be available.

During most of the mission, the reaction wheels are used for attitude control. Secular momentum growth due to solar pressure would be controlled using the thrusters. Momentum unloading could be autonomous or ground commanded. During orbit-adjust modes the thrusters would be used for attitude control since the disturbances due to the orbit-adjust thrusters would quickly saturate the reaction wheels.

26.5. Control-system design

The control system employs tachometer inner loops for the three reaction wheels. The outer loops are two proportional-integral-differential (PID) loops for the transverse axes and a proportional-integral (PI) loop for the spin-axis. The integral term in the PID controllers ensures accurate tracking of the spin-axis target. The inner momentum-wheel loops are PI controllers. A reaction wheel has a current feedback loop to counteract the back-electromotive force. This gives it a linear voltage to torque relationship. A momentum wheel does not have current feedback so as the wheel speed increases, the wheel will produce less torque. Using an inner loop solves this problem. The outer loops output an angular-acceleration demand that is converted to a wheel-speed demand and passed to the reaction-wheel tachometer loops. The inner loops ensure proper reaction-wheel response regardless of bearing friction magnitude, back-electromotive force, and model uncertainty. A sampling rate of 4 Hz was chosen for the digital implementation of this control system.

Fig. 26.2 shows the system. PID loops control roll and pitch that determine the pointing direction of the spin-axis. A PI loop controls the spin rate. The torque demand from the PID and PID are converted to a wheel-speed demand, as explained in the next section. The tachometer loop controls the wheel speed and this provides the torque on the spacecraft.

26.6. Nested loops

It is easiest to understand how the inner loop works with the outer loop if we look at the control of a single axis of the spacecraft.

Assume the dynamical system is composed of two equations for the pitch axis of a momentum bias spacecraft.

$$T = I\ddot{\theta} + J\dot{\Omega} \tag{26.1}$$

$$T_w = J\dot{\Omega} \tag{26.2}$$

where I is the pitch inertia, J is the momentum-wheel inertia, T is an external disturbance torque, θ is the angle that you want to control, Ω is your momentum wheel speed, and T_w is the torque on the wheel produced by an electric motor connecting the

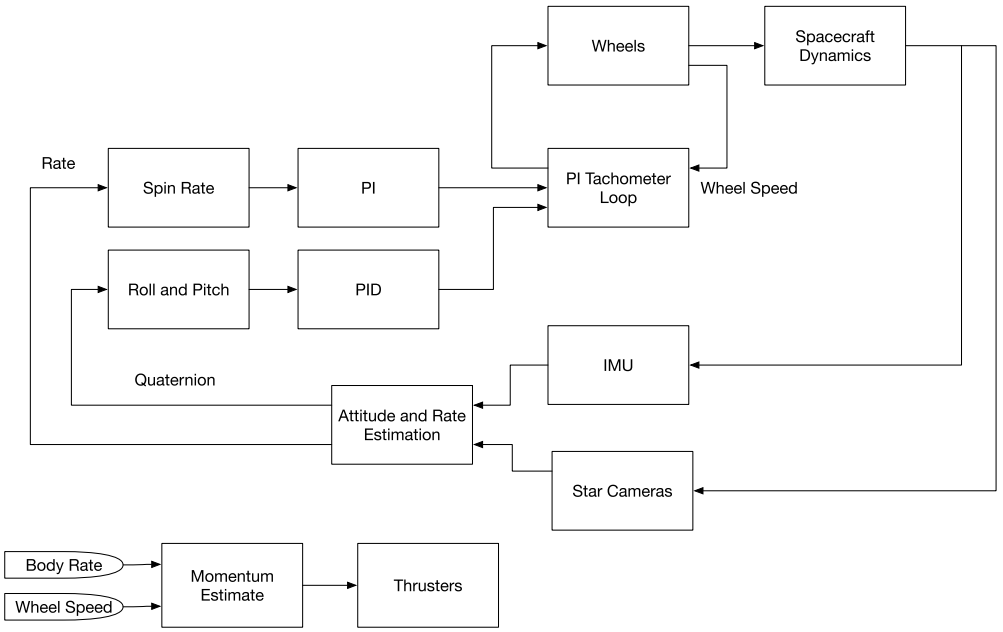


Figure 26.2 Control-system block diagram.

wheel to the spacecraft. You have a measurement of Ω and θ but not T_w. How do we connect the two? Write

$$J\dot{\Omega} = I\left(2\zeta\sigma\dot{\theta} + \sigma^2\theta\right) \tag{26.3}$$

where ζ is the damping ratio and σ is the undamped natural frequency. Our first dynamical equation is now

$$T = I\ddot{\theta} + I\left(2\zeta\sigma\dot{\theta} + \sigma^2\theta\right) \tag{26.4}$$

which is a damped second-order system. Divide by I to clarify the equation

$$\frac{T}{I} = \ddot{\theta} + \zeta\sigma\dot{\theta} + \sigma^2\theta \tag{26.5}$$

Our inner loop commands wheel speed,

$$T_w = Jk\left(\Omega_c - \Omega\right) \tag{26.6}$$

where k is the gain and Ω_c is the commanded wheel speed. The inner-loop dynamical equation is now

$$k\Omega_c = \dot{\Omega} + k\Omega \tag{26.7}$$

which is a damped first-order system. If k is high enough, that is the inner loop is very fast, the wheel speed will equal the commanded speed.

$$\Omega = \Omega_c \tag{26.8}$$

This is the "steady-state" response. How do we determine Ω? Integrate Eq. (26.3)

$$\Omega = \left(\frac{I}{J}\right) \int \left(2\zeta\sigma\dot{\theta} + \sigma^2\theta\right) \tag{26.9}$$

or

$$\Omega_c = \left(\frac{I}{J}\right) \sigma \left(2\zeta\dot{\theta} + \sigma \int \theta\right) \tag{26.10}$$

Our outer loop will control θ. Our inner loop will control Ω.

26.7. Simulation results

Acquisition of an inertial pointing vector was simulated. The mass properties and the actuator characteristics for the baseline MAP spacecraft were not available so generic parameters were used in the simulation. The spacecraft model is for a gyrostat with three reaction wheels. All nonlinear terms are included. The simulation models do not include sensor noise or external disturbances. The spacecraft is initially spinning at 0.464 rpm with the spin-axis aligned with the inertial Z-axis. The desired precession angle command of 22.5 deg is fed into the control loops through a low-pass filter to eliminate transients. The gains of the filter are chosen so that acquisition is completed within ten minutes.

Example 26.1 designs the control system. It creates three controllers

1. Two PIDs for roll and pitch;
2. One PI for yaw rate control;
3. One PI for the momentum-wheel tachometer loops.

The functions in the script use analytic pole placement. The PI and PID have rate filters so that at high frequencies they do not differentiate the incoming signals. This acts as a noise filter.

As can be seen from the plots, the control system successfully acquires and tracks the target. The overall pointing accuracy is better than 0.2 deg. The second plot shows the unit vector for the spin-axis of the spacecraft in three-dimensional space. Initially, it is aligned with the z-axis so its x and y components are zero. After the acquisition, the x- and y-axes trace out a circle in the XY-plane and the z value is constant.

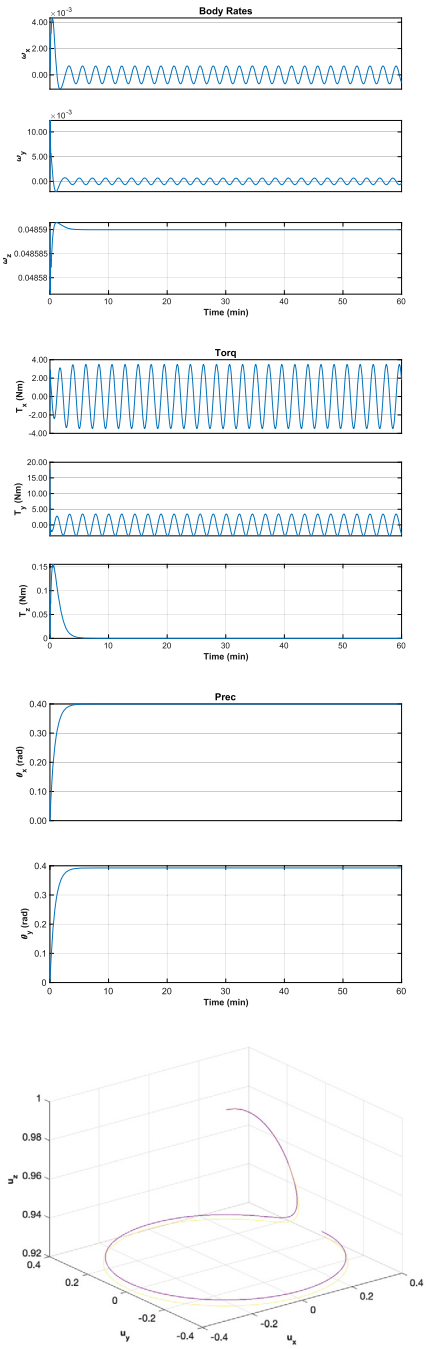
The control system presented here is only part of the overall control architecture that includes not only the other modes but also the attitude determination function, the momentum-unloading system, and the telemetry and command functions. Although

the spacecraft itself is mostly single string, a fault-detection and isolation system might also be included to protect the spacecraft against transient problems.

```
rPMToRPS     = pi/30;
tSamp        = 0.25; % Sampling time
dRHS         = RHSGyrostat;
dRHS.inr     = [2000,0,0;0,2000,0;0,0,4000];
dRHS.inrWheel = 1;

% RWA Tach loops
zeta     = 0.7071;   % Damping ratio
wN       = 1.0;  % Closed loop undamped frequency
[aTL,bTL,cTL,dTL] = PIDesign( zeta, wN, dRHS.
       inrWheel, tSamp, 'Delta' );

% Attitude Loops
zeta     = 0.7071;   % Damping ratio
wN     = 0.5;       % Closed loop undamped
       frequency
wR     = 4.0;       % Rate filter break frequency
tau    =  50;       % Integrator time constant
[aRoll ,bRoll, cRoll, dRoll]  = PIDMIMO( dRHS.inr
       (1,1), zeta, wN, tau, wR, tSamp, 'Delta');
[aPitch,bPitch,cPitch,dPitch] = PIDMIMO( dRHS.inr
       (2,2), zeta, wN, tau, wR, tSamp, 'Delta');

% Rate Loop
zeta         = 1.0;  % Damping ratio
wN           = 0.5;  % Closed loop undamped
       frequency
[aYaw,  bYaw,  cYaw,  dYaw]  = PIDesign( zeta, wN
       , dRHS.inr(3,3), tSamp, 'Delta' );

%% Initialize the control system
xTL              = zeros(3,1);
xRoll            = [0;0];
xPitch           = [0;0];
xYaw             = 0;
spinRate         = 0.464*rPMToRPS;
precRate         = 2*pi/3600;
precAngleDemand  = 22.5;
precAngleGain    = 0.995;
kMM              = 0.00;

% The control distribution matrix converts
% torque demand to angular acceleration demand
aRWA             = eye(3)/dRHS.inrWheel;
```

Example 26.1: Control-loop design code. This sets up the parameters that are used in the simulation.

```
nSim     = 4*3600;
tDist    = [0;0;0];
deg2R = pi/180;

% Initialize controllers
wRWAC          = [0;0;0];
tC             = [0;0;0];
precAngle  = 0;

% [q;omega;omegaRWA]
x  = [[1;0;0;0];[0;0;spinRate];[0;0;0]];
xP = zeros(29,nSim);
t  = 0;
for k = 1:nSim
  qIToB    = x(1:4);
  wCore    = x(5:7);
  wTach    = x(8:10);
  xI       = QTForm(qIToB,[0;0;1]);

  % Momentum Management
  hB       = dRHS.inr*wCore...
           + (dRHS.uWheel.*dRHS.inrWheel')*(wTach+
                 wCore);
  hTotal = QTForm( qIToB, hB );

  % Proportional controller for momentum
  tMM = QForm( qIToB, -kMM*hTotal );

  % Compute the Errors
  precAngle  = precAngleGain*precAngle + (1-
         precAngleGain)*precAngleDemand;
  cP = cos(precAngle*deg2R);
  sP = sin(precAngle*deg2R);
  xT = [sP*cos(precRate*t);sP*sin(precRate*t);cP
       ];
  xTB    = QForm( qIToB, xT );
  rollErr  =  asin(xTB(2));
  pitchErr = -asin(xTB(1));
  yawErr   = wCore(3) - spinRate;

  % The attitude control loops
  xRoll  =  xRoll + aRoll*xRoll + bRoll*rollErr;
  xPitch =  xPitch + aPitch*xPitch + bPitch*
         pitchErr;
  xYaw   =  xYaw + aYaw*xYaw + bYaw*yawErr;
  tC     = -[cRoll*xRoll + dRoll*rollErr;...
            cPitch*xPitch + dPitch*pitchErr;...
            cYaw*xYaw + dYaw*yawErr];

  % Convert torque demand to wheel speed demand
  wDRWA  = -aRWA*tC;

  % Integrate to get wheel speed demand
  wRWAC  = wRWAC + tSamp*wDRWA;

  % The RWA Tach Loops
  wErr = wTach - wRWAC;
  dRHS.torqueWheel = -dTL*wErr - cTL*xTL;
  xTL  = xTL + aTL*xTL + bTL*wErr;
  dRHS.torque = tMM + tDist;

  % Propagate
  x = RK4(@RHSGyrostat,x,tSamp,0,dRHS);
  t = t + tSamp;
  xP(:,k) = [x;tC;hTotal;tMM;...
    acos([xI(3);xT(3)]);...
    rollErr;pitchErr;xI;xT];
end

yL = [ dRHS.states(:)' ...
{'T_x␣(Nm)'} {'T_y␣(Nm)'}  {'T_z␣(Nm)'} ...
{'H_x␣(Nms)'} {'H_y␣(Nms)'} {'H_z␣(Nms)'} ...
{'T_x␣(Nm)'} {'T_y␣(Nm)'}  {'T_z␣(Nm)'}...
{'\theta_x␣(rad)'} {'\theta_y␣(rad)'}...
{'Roll␣(rad)'} {'Pitch␣(deg)'}...
{'u_x'} {'u_y'} {'u_z'} ];

t = (0:nSim-1)*tSamp; k = 1:4;
TimeHistory(t,xP(k,:),yL(k),'Quat');k = 5:7;
TimeHistory(t,xP(k,:),yL(k),'Body␣Rates');k=k+3;
TimeHistory(t,xP(k,:),yL(k),'RWA');k=k+3;
TimeHistory(t,xP(k,:),yL(k),'Torq');k=k+3;
TimeHistory(t,xP(k,:),yL(k),'Momentum');k=k+3;
TimeHistory(t,xP(k,:),yL(k),'MM␣Torq');k=[20 21];
TimeHistory(t,xP(k,:),yL(k),'Prec');k=[22 23];
TimeHistory(t,xP(k,:),yL(k),'Errors');

j1 = 24:26; j2 = 27:29;
PlotV([xP(j1,:);xP(j2,:)],yL{j1},'Spin␣Axis')

Figui
```



Example 26.2: Spin up and acquisition simulation. The spacecraft acquires the target quickly.

References

[1] L. Markley, S. Andrews, J. O'Donnell, D. Ward, A. Ericsson, The Microwave Anisotropy Probe Attitude Control System, 10 2011.

CHAPTER 27

Solar sails

27.1. Introduction

Solar sails are a potential method for traveling within the solar system using the Sun as the propulsion source. The advantage is that no fuel is carried, thus permitting large velocity changes. There are many applications of solar sails. For example, a solar-sail could be sent out of the solar system by first diving at the Sun, passing around the Sun quickly and then heading on a path at very high velocity into deep space. Another application for solar sails is known as a pole sitter where the sail force cancels a planet's and the Sun's gravitational attraction and allows it to be stationary above a planet's pole [1]. This would allow continuous observation of a planet from the polar point of view.

An example of a solar sail is the NASA Near-Earth Asteroid Scout, or NEA Scout, [2,3] is a mission to map an asteroid and demonstrate several technological firsts, including being the first CubeSat to reach an asteroid. NEA Scout is designed to perform reconnaissance of an asteroid using a CubeSat and solar-sail propulsion, which offers navigational agility during the cruise for approaching the target. Using the Sun to propel the spacecraft, NEA Scout (Fig. 27.1) was designed to fly past the 92-m diameter asteroid. NEA Scout could track the asteroid's orbit and measure its shape, rotational properties, spectral class, local dust and debris field, regional morphology, and regolith properties.

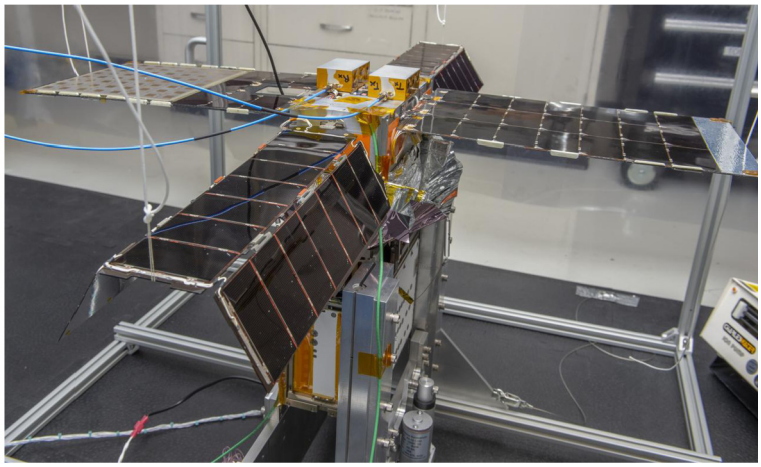


Figure 27.1 The NEA Scout solar-sail spacecraft is shown during integration and test. Image courtesy of NASA.

ADCS - Spacecraft Attitude Determination and Control
https://doi.org/10.1016/B978-0-32-399915-1.00039-5

This chapter describes the design of a solar-sail attitude control system. Fig. 27.2 shows some types of sails. From left, clockwise, control is accomplished by

- Moving the center-of-mass;
- Rotating the vanes to produce control torques;
- Rotating the sail segments to produce control torques;
- Rotating the boom to move the center-of-mass.

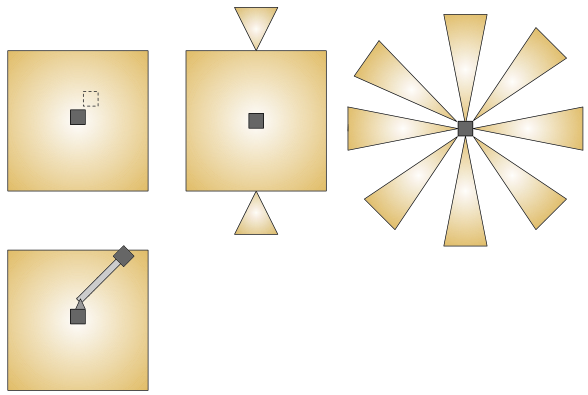


Figure 27.2 Four solar-sail types.

27.2. Gyrostat with a moving mass

Reaction wheels can be used to control a solar sail. A spacecraft with reaction wheels is modeled as a gyrostat, which is a rigid spacecraft, except for symmetric rotors that spin about axes fixed to the body frame. Many spacecraft are gyrostats since reaction wheels can control the spacecraft's attitude without the need to expend propellant. A gyrostat is used to represent spacecraft with reaction wheels. The reaction wheels would quickly saturate if a momentum-control method is not included. In this control system, a moving mass is used to manage momentum.

The momentum for a gyrostat is

$$H = A\left(I\omega + \sum_k u_k J_k(\Omega_k + u_k^T\omega)\right) \tag{27.1}$$

where Ω_k is the angular rate of the kth wheel about its spin-axis, ω are the body rates, J_k is the spin-axis moment of inertia, u_k is the unit vector of the wheel spin-axis measured in the body frame, and I is the inertia of the rigid body excluding the wheels.

The equations of motion are then

$$T = I\dot{\omega} + \omega^{\times}\left(I\omega + \sum_k u_k J_k(\Omega_k + u_k^T\omega)\right) + \sum_k u_k J_k(\dot{\Omega}_k + u_k^T\dot{\omega}) \tag{27.2}$$

$$T_k = J_k(\dot{\Omega}_k + u_k^T\dot{\omega}) \tag{27.3}$$

The transverse axes' inertias are lumped into the core body inertia. T is the body-fixed torque on the spacecraft and T_k is the scalar torque on each wheel. These equations give $3 + n$ equations of motion. To integrate these equations, you need first to solve for the angular acceleration of the core, and then substitute the angular-acceleration vector into the reaction-wheel equations,

$$\dot{\omega} = I^{-1}\left(T - \omega^{\times}\left(I\omega + \sum_k u_k J_k(\Omega_k + u_k^T\omega)\right) - \sum_k u_k T_k\right) \tag{27.4}$$

$$\dot{\Omega}_k = \frac{T_k}{J_k} - u_k^T\dot{\omega} \tag{27.5}$$

Momentum is controlled using a moving mass. The moving-mass model is depicted in Fig. 27.3. The first image shows that the model has two sliders, each with one degree-of-freedom (e.g., body 2 is fixed to stepper 2), as well as the track geometry for the NEAR Scout spacecraft. The second figure shows the dynamic quantities.

Table 27.1 gives the definitions of the variables. The core body, 1, has the reaction wheels.

Table 27.1 Moving-mass variables.

Variable	Definition	Constant?
B_1	Transformation matrix from slider 1 to the core body	Yes
B_2	Transformation matrix from slider 2 to slider 1	Yes
B	Transformation matrix from body 2 to body 1	No
I_1	Body 1 inertia matrix	Yes
I_2	Body 2 inertia matrix	Yes
m_1	Body 1 mass	Yes
m_2	Body 2 mass	Yes
A	Transformation matrix from body 1 to the inertial frame (ECI)	No
ω_1	Angular rate vector of body 1 with respect to the ECI frame measured in the body-1 frame	No
ω_2	Total angular rate vector of body 2 with respect to the ECI frame measured in the body-2 frame	No
Ω_2	Total angular rate vector of body 2 with respect to body 2 measured in the body 2 frame	No

continued on next page

Table 27.1 (*continued*)

Variable	Definition	Constant?
D_1	Vector from the total center-of-mass to the center-of-mass of body 1 written in the ECI frame	No
D_2	Vector from the total center-of-mass to the center-of-mass of body 2 written in the ECI frame	No
d_1	Vector from the total center-of-mass to the center-of-mass of body 1 written in the body 1 frame	No
d_2	Vector from the total center-of-mass to the center of mass of body 2 written in the body 1 frame	No
r_1	Vector from the reference point of body 1 written in the body-1 frame	No
r_2	Vector from the reference point of body 1 to body 2 written in the body-1 frame	No
u_1	Slider 1 unit vector	Yes
u_2	Slider 2 unit vector	Yes
ρ_1	Distance along slider 1 unit vector	No
ρ_2	Distance along slider 2 unit vector	No
λ_1	Distance from reference to slider 1 origin	Yes
λ_2	Distance from slider 2 tip to center-of-mass of body 2	Yes
c	Center-of-mass in the body 1 frame	No

Fig. 27.3 shows the two bodies that are connected via two sliders. Each slider has a single degree-of-freedom that we define as always along the z-axis of the slider. The transformation matrix from slider 1 to the core is B_1 and from slider 2 to slider 1 is B_2. r goes from the origin to the mass. X is from the inertial origin to the spacecraft center-of-mass (which may move). d goes from the spacecraft center-of-mass to the moveable mass.

The total angular momentum of the vehicle is

$$H = AI_1\omega_1 + ABI_2\omega_2 + m_1 D_1^{\times}\dot{D}_1 + m_2 D_2^{\times}\dot{D}_2 \tag{27.6}$$

where I_1 is the inertia with respect to the center-of-mass of the core, I_2 is the inertia matrix with respect to the center-of-mass of body 2, D_k are in the inertial frame, and A is the transformation matrix from the body to the inertial frame. $B = B_1 B_2$.

The notation $a^{\times}$ is the matrix equivalent of the crossproduct

$$a^{\times} = \begin{bmatrix} 0 & -a_x & a_y \\ a_x & 0 & -a_z \\ -a_y & a_z & 0 \end{bmatrix} \tag{27.7}$$

$$D_1 = Ad_1 \tag{27.8}$$

$$D_2 = ABd_2 \tag{27.9}$$

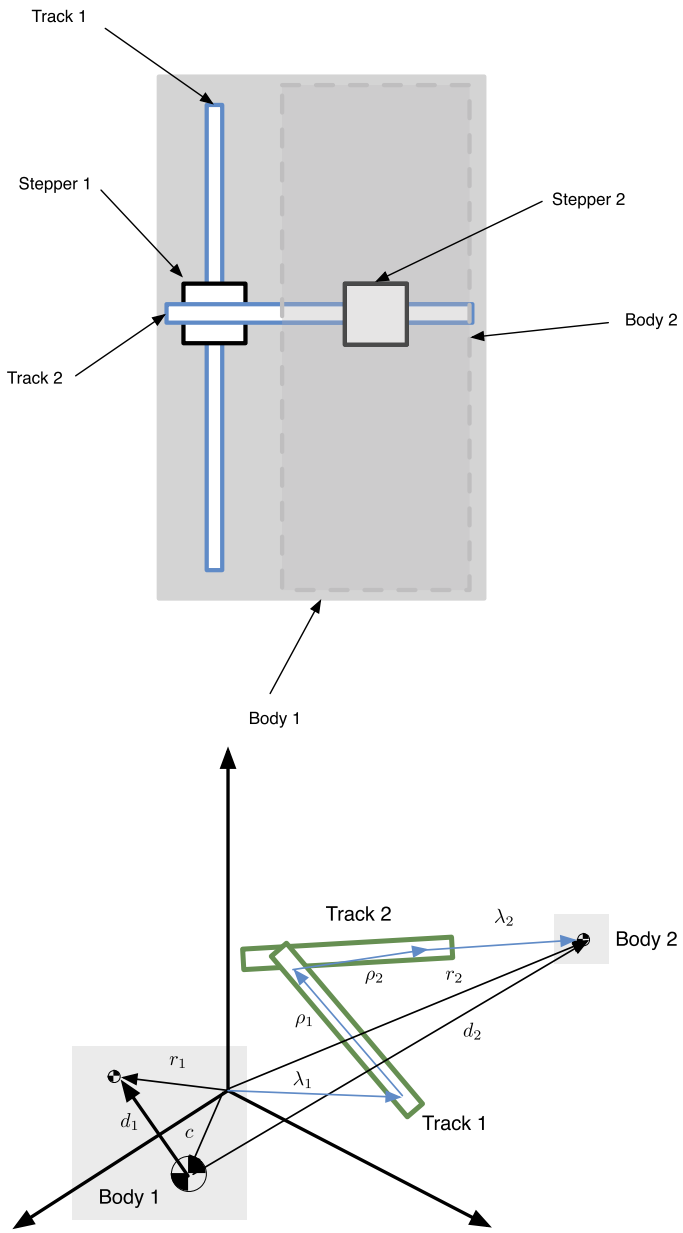


Figure 27.3 Moving-mass geometry.

where B is a constant and

$$d_k = r_k - c \tag{27.10}$$

where c is the location of the center-of-mass and r_1 is a constant,

$$r_2 = \lambda_1 + B_1 u_1 \rho_1 + B_1 B_2 (u_2 \rho_2 + \lambda_2) \tag{27.11}$$

and ρ_k is a scalar. λ_1 and λ_2 are constants.

The center-of-mass c is

$$c = \frac{1}{M} (m_1 r_1 + m_2 r_2) \tag{27.12}$$

where M is the total mass, or

$$c = \mu_1 r_1 + \mu_2 r_2 \tag{27.13}$$

where $\mu_k = \frac{m_k}{M}$.

We can then write

$$d_1 = r_1 - c = r_1(1 - \mu_1) - \mu_2 r_2 \tag{27.14}$$
$$d_1 = \mu_2 (r_1 - r_2) \tag{27.15}$$
$$d_2 = \mu_1 (r_2 - r_1) \tag{27.16}$$

The derivatives are

$$\dot{d}_1 = -\mu_2 \dot{r}_2 \tag{27.17}$$
$$\dot{d}_2 = \mu_1 \dot{r}_2 \tag{27.18}$$
$$\dot{r}_2 = B_1 (u_1 \dot{\rho}_1 + B_2 u_2 \dot{\rho}_2) \tag{27.19}$$

The accelerations are

$$\ddot{d}_1 = -\mu_2 \ddot{r}_2 \tag{27.20}$$
$$\ddot{d}_2 = \mu_1 \ddot{r}_2 \tag{27.21}$$
$$\ddot{r}_2 = B_1 (u_1 \ddot{\rho}_1 + B_2 u_2 \ddot{\rho}_2) \tag{27.22}$$

$$\dot{D}_1 = A \left(\omega^{\times} d_1 + \dot{d}_1 \right) \tag{27.23}$$
$$\dot{D}_2 = A \left(\omega^{\times} d_2 + \dot{d}_2 \right) \tag{27.24}$$
$$\ddot{D}_1 = A \left(\omega^{\times} \omega^{\times} d_1 + 2\omega^{\times} \dot{d}_1 + \ddot{d}_1 - d_1^{\times} \dot{\omega} \right) \tag{27.25}$$
$$\ddot{D}_2 = A \left(\omega^{\times} \omega^{\times} d_2 + 2\omega^{\times} \dot{d}_2 + \ddot{d}_2 - d_2^{\times} \dot{\omega} \right) \tag{27.26}$$

We need to find the three core dynamical equations for the entire spacecraft and the two scalar equations for the masses. Taking the derivative of Eq. (27.6) we get

$$T = I\dot{\omega} + \omega^{\times} I\omega + m_1 d_1^{\times}\left(\omega^{\times}\omega^{\times} d_1 + 2\omega^{\times}\dot{d}_1 + \ddot{d}_1 - d_1^{\times}\dot{\omega}\right) \tag{27.27}$$
$$+ m_2 d_2^{\times}\left(\omega^{\times}\omega^{\times} d_2 + 2\omega^{\times}\dot{d}_2 + \ddot{d}_2 - d_2^{\times}\dot{\omega}\right)$$

where $I = I_1 + BI_2B^T$, $\omega = \omega_1$ and using the identity

$$\left(Ad_k\right)^{\times} = Ad_k^{\times}A^T \tag{27.28}$$

The final step is to get the scalar equations for the movement of the masses. We write

$$Au_1 f_1 = m_1\ddot{D}_1 + m_2\ddot{D}_2 \tag{27.29}$$
$$Au_2 f_2 = m_k\ddot{D}_2 \tag{27.30}$$

For the outer link

$$u_k f_2 = m_2\left(\omega^{\times}\omega^{\times} d_2 + 2\omega^{\times}\dot{d}_2 + \ddot{d}_2 - d_2^{\times}\dot{\omega}\right) \tag{27.31}$$

Multiplying both sides by u_2^T

$$f_2 = m_2 u_2^T\left(\omega^{\times}\omega^{\times} d_2 + 2\omega^{\times}\dot{d}_2 + \ddot{d}_2 - d_2^{\times}\dot{\omega}\right) \tag{27.32}$$

which is a scalar and

$$T = I\dot{\omega} + \omega^{\times} I\omega + m_1 d_1^{\times} u_1 f_1 + m_2 d_2^{\times}\ u_2 f_2 \tag{27.33}$$

27.3. Thin-membrane model

Since the sail is a thin membrane, we can make the additional assumption that it is at a constant temperature, therefore we can perform a flux balance and combine the optical and thermal forces into one model. The flux balance is illustrated in Fig. 27.4. Each side of the membrane, front and back, has its emissivity (ϵ).

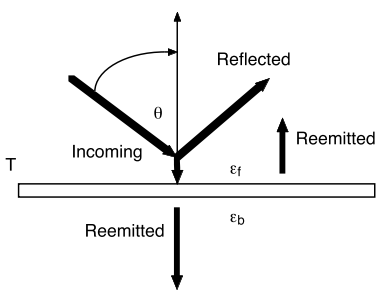


Figure 27.4 Flux balance on the solar-sail surface.

In steady state, the incoming absorbed solar flux must equal the outgoing reemitted flux, which follows Boltzmann's law for thermal radiation. Table 27.2 gives the definitions of each of the variables used:

Table 27.2 Solar Sail Variables.

Variable	Definition	Constant
ρ_a	Absorption coefficient	Yes
P	Incoming solar flux, $P = \Phi \cos(\theta)$	No
σ	Stefan–Boltzmann constant (5.670×10^{-8} J K^{-4} s^{-1})	Yes
ϵ_f	Emissivity of front of sail	Yes
ϵ_b	Emissivity of back of sail	Yes
T_s	Temperature of the front and back sails (assumed to be equal)	No

We can write:

$$\rho_a P = \sigma \left(\epsilon_f T_s^4 + \epsilon_b T_s^4\right) \tag{27.34}$$

We can solve for the equilibrium temperature as:

$$T_s = \left(\frac{\rho_a P}{\sigma(\epsilon_f + \epsilon_b)}\right)^{1/4} \tag{27.35}$$

The reemitted fluxes per unit area for the front and back of the membrane are independent of temperature:

$$\Phi_f = \frac{\epsilon_f \rho_a P}{\epsilon_f + \epsilon_b} \tag{27.36}$$

$$\Phi_b = \frac{\epsilon_b \rho_a P}{\epsilon_f + \epsilon_b} \tag{27.37}$$

The remaining outgoing fluxes accounting for the specular and diffuse reflected portions of the solar flux are:

$$\Phi_s = \rho_s P \tag{27.38}$$

$$\Phi_d = \rho_d P \tag{27.39}$$

From the conservation of energy, we know that these four outgoing fluxes must equal the incoming flux. The total force exerted on the sail is:

$$F = \cos(\theta)\left(f_s + f_d + f_f + f_b\right) \tag{27.40}$$

where the normalized force contributions due to each flux are:

$$f_s = -2\cos(\theta)\Phi_s \hat{n} \tag{27.41}$$

$$
\begin{aligned}
f_d &= -\Phi_d \left(\frac{2}{3}\hat{n} + \hat{s} \right) \\
f_f &= -\frac{2}{3}\Phi_f \hat{n} + \Phi_f \hat{s} \\
f_b &= \frac{2}{3}\Phi_b \hat{n} + \Phi_b \hat{s}
\end{aligned}
\tag{27.42}
$$

Combining terms, we get the final form of the thermal/optical force model:

$$
F = -pA\cos(\theta)\left(2\left[\rho_s \cos(\theta) + \frac{1}{3}\left(\rho_d + \rho_a \frac{\epsilon_f - \epsilon_b}{\epsilon_f + \epsilon_b}\right)\right]\hat{n} + [\rho_d + \rho_a]\hat{s}\right) \tag{27.43}
$$

To express the force in terms of components normal and tangential to the sail, we note that the Sun unit vector can be expressed as $\hat{s} = \cos(\theta)\hat{n} + \sin(\theta)\hat{t}$. Therefore,

$$
f_n = pA\cos(\theta)\left((1+\rho_s)\cos(\theta) + \frac{2}{3}\left(\rho_d + \rho_a \frac{\epsilon_f - \epsilon_b}{\epsilon_f + \epsilon_b}\right)\right) \tag{27.44}
$$

$$
f_t = pA\cos(\theta)\sin(\theta)(\rho_d + \rho_a) \tag{27.45}
$$

27.4. Momentum control

Momentum control is done either with thrusters or with the moving-mass system.

The momentum-control equation is a proportional-integral (PI) controller.

$$
\dot{h} = -K\left(h + \frac{1}{\tau}\int h\right) \tag{27.46}
$$

where the momentum h and torque T are in the inertial frames. The gain and integral time constant are set to maintain wheel margins and provide damping. When thrusters are used the computation of torque is straightforward.

When the moving mass is used we have the equation for the body-frame torque due to the moving mass

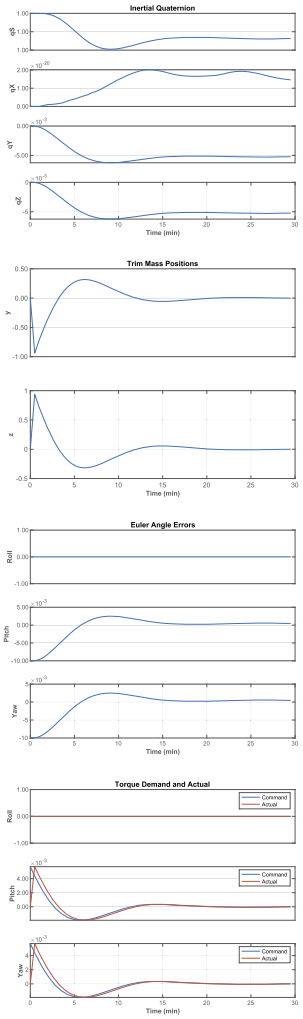
$$
T = \left(c_p - c(\rho)\right)^{\times} F \tag{27.47}
$$

where F is the sail force in the body frame. We only have control over two elements of ρ.

We solve this by using the MATLAB® function fminsearch to find the two positions that best fit the torque using the center-of-mass calculation in Eq. (27.12). fminsearch does not apply constraints. If the demand on ρ is at a limit it is set to a limit.

A standalone momentum-control demo shows the results given in Example 27.1.

```
% Constants
clear d
c          = 3e8;
pSun       = 1367;
wSail      = 100;
%  User Parameters
pitchStep  = 0.01;  % y-axis, radians
yawStep    = 0.01;  % z-axis, radians
g = load('PlateWithMasses');

%% Design controller
xN             = zeros(6,1);
iner           = diag([1 1 1]);
zeta           = 2*ones(1,3);
wn             = 0.001*ones(1,3);
tauInt         = 5000*ones(1,3);
omegaR         = 5*wn;
dT             = 30;
[aC,bC,cC,dC] = PIDMIMO( iner, zeta,...
  wn, tauInt, omegaR, dT);
uControl = [0 1 0; 0 0 1]';
dOffset  = zeros(3,3);
mControl = [g.body(1).mass.mass g.body(2).mass.
      mass g.body(3).mass.mass];
% States for each body, including attitude, are
      stacked
xCore = [zeros(6,1);QZero;zeros(3,1)];
xMass = [zeros(6,1);QZero;zeros(3,1)];
x     = [xCore; xMass; xMass];
% Indices
iR1    = 2+13;  % y mass position
iR2    = 3+26;  % z mass position
iV1    = 5+13;  % y mass velocity
iV2    = 6+26;  % z mass velocity
% New attitude command
angCommand = [0;pitchStep;yawStep];
% Assume a constant sail force in the ECI frame
f          = struct;
f.total    = [-2*pSun/c*wSail^2;0;0];
tq         = struct;
tq.total = [0;0;0];
% Center of pressure
Cp         = [0;0;0];
% Additional fields for RHS
d.g        = g;
d.nBody    = 3;
maxRate    = 1;  % m/s
% Number of simulation points
nSim = 60;
xP  = zeros(size(x,1)+9,nSim);

% Simulation loop
for k = 1:nSim
  % Transverse angle error (small angles)
  u            = QTForm( x(7:10), [1;0;0] );
  angleError = [0;-u(3)-angCommand(2);u(2)-
      angCommand(3)];
  % Control
  yN = cC*xN + dC*angleError;
  xN = aC*xN + bC*angleError;
  Tcommand = -g.mass.inertia*yN;
  cM = TorqueToCM( Tcommand, f.total, Cp );
  rhoCommand = CMToMassPositions( cM, mControl,
      dOffset, uControl );
  rhoActual = [x(iR1); x(iR2)];
  deltaRho = rhoCommand - rhoActual;
  rhoDot = sign(deltaRho).*min( abs(deltaRho)/dT,
      maxRate*[1;1] );
  % Update rates
  xRates = x;
  xRates(iV1) = rhoDot(1);
  xRates(iV2) = rhoDot(2);
  xNew = FMovingBody( 'init', x, xRates, tq,
      struct('g', g) );
  % Disturbances - simple torque model
  g.body(2).rHinge = [0;rhoActual(1);0];
  g.body(3).rHinge = [0;0;rhoActual(2)];
  cMActual = (mControl(2)*g.body(2).rHinge +
      mControl(3)*g.body(3).rHinge)/g.mass.
      mass;
  tq.total = Cross( Cp - cMActual, f.total );
  d.force  = f;
  d.torque = tq;
  % Store
  xP(:,k)  = [x; angleError; Tcommand; tq.total];
  % Integrate
  x = RK4( @FCoreAndMoving, xNew, dT, 0, '', d );
end

%% Plot
t = (0:nSim-1)*dT;
rPY = {'Roll','Pitch','Yaw'};
TimeHistory(t,xP(7:10,:),{'qS','qX','qY','qZ'},'
      Quaternion' );
TimeHistory(t,xP([iR1 iR2],:), {'y','z'},'Trim
      Mass' );
TimeHistory(t,xP(40:42,:),rPY,'Angle Errors');
l = {'Command' 'Actual'};
TimeHistory( t, xP(43:48,:), rPY,...
  'Torque Demand and Actual',{[1 4],[2 5],[3
      6]},{l l l});
Figui
```



Example 27.1: Control with moving masses.

27.5. Attitude control

This attitude control system example uses a proportional-integral-differential (PID) controller with reaction wheels for control and vanes for momentum unloading. Vanes are placed at the corners of the sail and have a single axis of rotational freedom. The configuration is shown in Fig. 27.5.

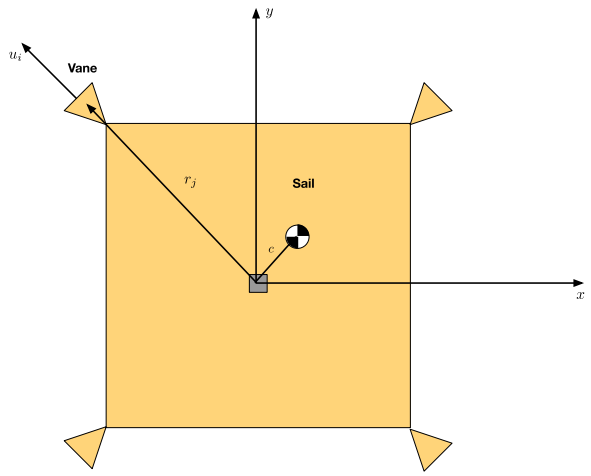


Figure 27.5 Sail with vanes. c is the vector from the geometric center to the center-of-mass. u_i is the rotation axis for the vanes.

The core control system will use a PID controller.

$$\frac{u_c(s)}{\theta(s)} = K_p\left(1 + \tau_d s + \frac{1}{\tau_i s}\right) \tag{27.48}$$

where τ_d is the rate time constant, which is how long the system will take to damp and τ_i is how fast the system will cancel a steady disturbance. The controller will drive the reaction wheels. The momentum controller will use a proportional-integral (PI) controller.

$$\frac{T_c(s)}{H(s)} = K_p\left(1 + \frac{1}{\tau_i s}\right) \tag{27.49}$$

The controller will command vane angles.

The torque on the sail and the vanes, assuming perfect specular reflection is

$$t = -2p\sum_{i=1}^{N} a_i \left(s^T n_i\right)^2 (r_i - c)^{\times} n_i \tag{27.50}$$

where p is the solar pressure, which is the solar flux divided by the speed-of-light, a_i is the area of each surface, and n_i is the normal for each surface. The normal vector for

the main sail is along $+z$. The normal vectors for a vane, as a function of vane angle is

$$n_i = \begin{bmatrix} \cos\alpha_i & \sin\alpha_i & 0 \\ -\sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta_i & 0 & \sin\beta_i \\ 0 & 1 & 0 \\ -\sin\beta_i & 0 & \cos\beta_i \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{27.51}$$

This simplifies to

$$n_i = \begin{bmatrix} \cos\alpha_i \sin\beta_i \\ -\sin\alpha_i \sin\beta_i \\ \cos\beta_i \end{bmatrix} \tag{27.52}$$

Linearized about small β_i,

$$n_i = \begin{bmatrix} \beta_i \cos\alpha_i \\ \beta_i \sin\alpha_i \\ 1 \end{bmatrix} \tag{27.53}$$

where β_i are the control angles.

The performance of vanes can be understood by looking at the torque-balancing problem. To unload momentum, the vanes need to be able to balance the net torque on the sail. Example 27.2 shows an example of torque balancing. The area of the vanes must

```
%% Momentum control with vanes
r       = 50; % m
d.r     = r*[-1  1 -1  1  0;...
              1  1 -1 -1  0;...
              0  0  0  0  0];
d.alpha = [pi/4  3*pi/4  5*pi/4  7*pi/4  0];
d.area  = [4  4  4  4  100*100];
d.c     = [0.02;0.2;0.1];
d.uSun  = [0;0;1];

beta = [0;0;0;0];
t0   = TorqueVane(beta,d);

opts = optimset('fminsearch');

beta = fminsearch(@FVanes,beta,[],d);
tF   = TorqueVane(beta,d);

disp('Torque')
fprintf('Initial [%8.1e %8.1e %8.1e] micro-Nm\n',
        t0*1e6);
fprintf('Optimal [%8.1e %8.1e %8.1e] micro-Nm\n',
        tF*1e6);
fprintf('Angles   [%5.2f %5.2f %5.2f %5.2f] deg\n'
        ,beta)*180/pi;

%% Function called by fminsearch
function y = FVanes(beta,d)

 t = TorqueVane(beta,d);
 y = Mag(t);

end
```

```
Torque
Initial [ 9.1e+03 -9.1e+02  0.0e+00] micro-Nm
Optimal [-2.6e-02 -5.9e-03 -7.0e-03] micro-Nm
Angles  [-0.93 -0.44  0.03  1.39] deg
```

Example 27.2: Vanes for torque balancing.

be sufficient to balance the torque due to the offset of the mainsail center-of-pressure from the spacecraft center-of-mass.

27.6. Architecture

The attitude control system is shown in Fig. 27.6. The blue boxes are actuator hardware. The green boxes are sensor hardware. The black boxes are included in the default attitude control system. The gray boxes, needed for a flight system, are not included. IMU stands for inertial measurement unit. It includes both gyros and linear accelerometers. For this application, only the gyros are used.

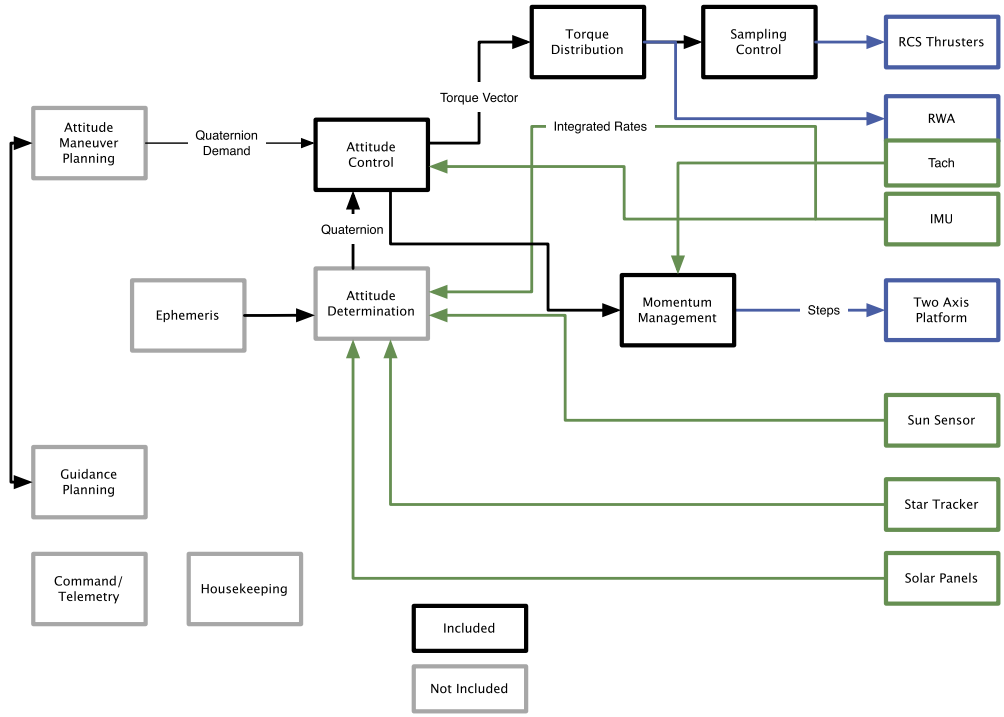
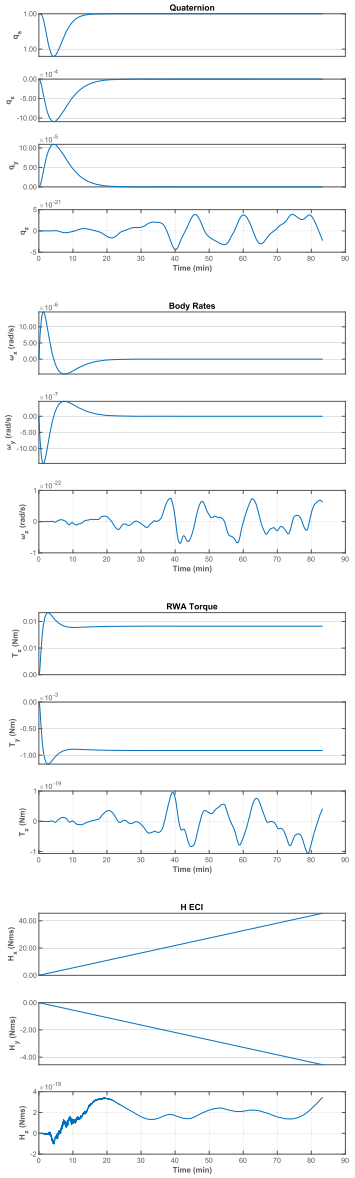


Figure 27.6 GN&C system. The black boxes are included ACS software. IMU stands for inertial measurement unit.

Example 27.3 shows the sail simulation. The true quaternion is used as the attitude reference. The attitude determination system is not simulated. Fig. 27.7 shows the simulation with the momentum-management system enabled. The algorithm for computing the β angles uses the last value as a starting point. This results in smooth operation. Since it quickly nulls the net torque it is constant. This assumes that it has a good model of the sail. In practice, there will be errors and some momentum will build up and have

```
%% Constants
controlOn = true;
mmOn      = true; % Momentum management
rToD      = 180/pi;

%% User initialization
n         = 5000; % Number of simulation steps
dT        = 1.0; % Time step (s)

% Momentum Controller
omegaC    = 0.1; % Control bandwidth
zetaC     = 1; % Control damping ratio

% Initial states
q         = [1;0;0;0];
omega     = [0;0;0];
omegaRWA  = [0;0;0];
uSun      = [0;0;1];
r         = 50; % m
dV.r      = r*[-1  1 -1  1  0;...
                1  1 -1 -1  0;...
                0  0  0  0  0];
dV.alpha  = [pi/4  3*pi/4  5*pi/4  7*pi/4  0];
dV.area   = [20  20  20  20  100*100];
dV.c      = [0.02;0.2;0.1];
dV.uSun   = uSun;

%% State vector
x     = [q;omega;omegaRWA];
dRHS  = RHSGyrostat;
dRHS.inr = diag([25000  25000  50000]);

%% Simulate
xP        = zeros(length(x)+13,n);
% Edit the control system data structure
dC        = SailACS;
dC.inr    = dRHS.inr;
dC.dT     = dT;
dC.omega  = omega; % The LVLH rate
dC.mmOn   = mmOn;
dC.kF     = dRHS.inr(2,2)*omegaC^2;
dC.tauR   = 2*omegaC*zetaC*dRHS.inr(2,2)/dC.kF;
dC.vanes  = dV;
dC.uSun   = uSun;

% Create a time vector
t  = (0:n-1)*dT;

% Simulation loop
for k = 1:n
  % Update the controller
  if( controlOn )
    [dC,beta,tRWA] = SailACS(x,dC);
  else
    beta = [0;0;0;0]; tRWA = [0;0;0];
  end
  dV.uSun = QForm(x(1:4),uSun);
  tSail = TorqueVane(beta,dV);

  dRHS.torque = tSail; % Nm
  dRHS.torqueWheel = tRWA;
  [~,hECI] = RHSGyrostat(x,0,dRHS);
  xP(:,k) = [x;tRWA;beta*rToD;hECI;tSail];
  x = RK4(@RHSGyrostat,x,dT,0,dRHS);
end

%% Plotting
yL = {'q_s' 'q_x' 'q_y' 'q_z' ...
'\omega_x (rad/s)' '\omega_y (rad/s)' '\omega_z (
      rad/s)' ...
'\Omega_1' '\Omega_2' '\Omega_3' ...
'T_x (Nm)' 'T_y (Nm)' 'T_z (Nm)' ...
'\beta_1 (deg)' '\beta_2 (deg)' '\beta_3 (deg)' '
      \beta_4 (deg)' ...
'H_x (Nms)' 'H_y (Nms)' 'H_z (Nms)' ...
'T_x (Nm)' 'T_y (Nm)' 'T_z (Nm)'};

tL = {'Quaternion' 'Body Rates' 'Wheel Rates' ...
      'RWA Torque' 'beta' 'H ECI' 'Sail' };
kL = {1:4 5:7 8:10 11:13 14:17 18:20 21:23};
for j = 1:length(tL)
  k = kL{j};
  TimeHistory(t,xP(k,:),yL(k),tL{j});
end

Figui
```



Example 27.3: Attitude control simulation with vanes for momentum control. Momentum control is not enabled.

to be removed. The control system does not feed forward the solar torque. The attitude controller is fast enough to make that unnecessary.

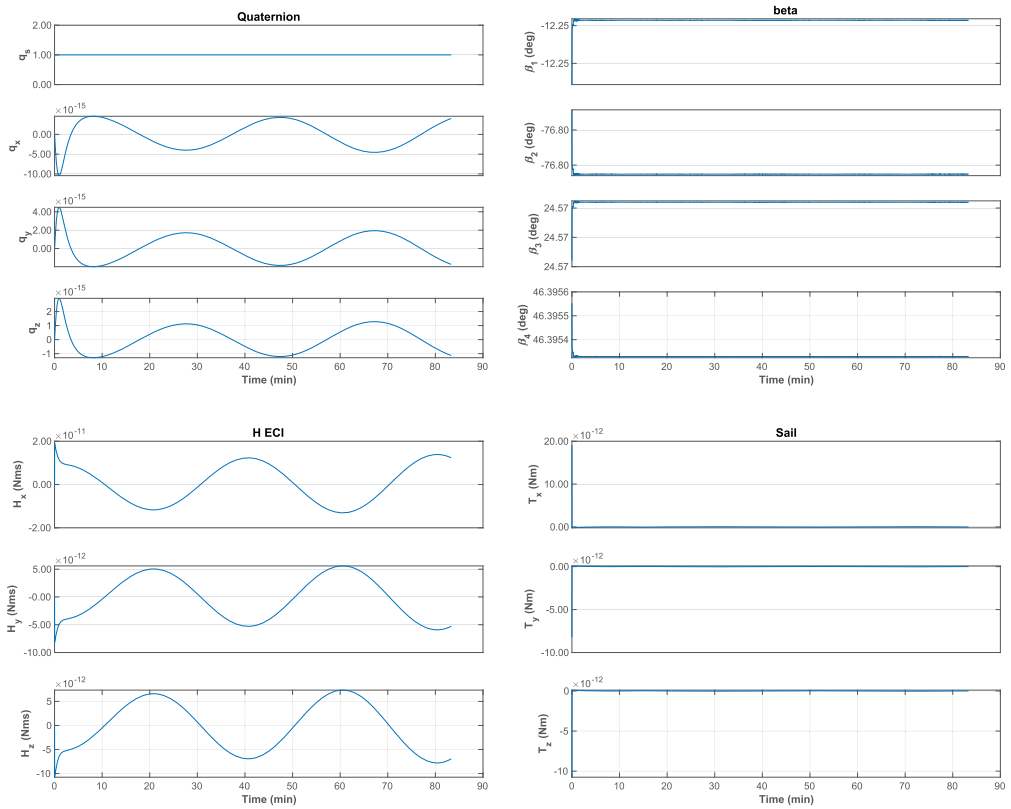


Figure 27.7 Attitude control simulation with vanes enabled for momentum control. The large angles necessitate the use of a numerical method to solve for the angles.

References

[1] J. Heiligers, M. Vergaaij, M. Ceriotti, End-to-end trajectory design for a solar-sail-only pole-sitter at Venus, Earth, and Mars, in: 5th International Symposium on Solar Sailing, ISSS 2019; Conference date: 30-07-2019 through 02-08-2019, Advances in Space Research 67 (9) (2021) 2995–3011.

[2] JPL, Near Earth Asteroid Scout (NEAScout), http://www.jpl.nasa.gov/cubesat/missions/neascout.php, 2016.

[3] A. Heaton, J. Castillo-Rogez, L. Johnson, L. Jones, Near Earth asteroid rendezvous mission, in: AIAA Space 2014, August 2014.

APPENDIX A

Math

This appendix provides an introduction to the mathematics used in spacecraft simulation and control-system design. The topic of numerical integration is covered because virtually all simulations of spacecraft require numerical integration. Matrices and vectors are important elements needed to model spacecraft. Spherical geometry is covered along with some elements of calculus.

A.1. Vectors and matrices

A.1.1 Notation

In this book, vectors may represent either ordered collections of values, or a quantity with a magnitude and direction in a 3-space. The state vector for a system is an example of the former; the position vector of a thruster is an example of the latter. The state vector contains all quantities to represent the time-varying quantities of a system. For a point mass these would be position and velocity. For a circuit it would be the currents in the wires.

This book uses a combination matrix and vector notation to represent vectors and matrices. Matrix notation implies that each vector is connected with a specific coordinate frame. Thus the position vector

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{A.1}$$

cannot be assigned a value unless the reference frame for which x, y, and z are defined is given. Usually the reference frame will be obvious from the context. For example, if the transformation matrix B transforms from frame b to a, then

$$y = Bu \tag{A.2}$$

implies that u is in frame b and y is in frame a. Note the use of a capital letter for the matrix. In some cases, mostly when presenting other authors' work, vector notation will be used and will be clear from the context. A vector, in contrast to a "3×1 matrix" exists independently of any coordinate frame conceptually, but when numbers are to be used the coordinate frame must be specified.

Subscripts will be used to denote the elements of a vector. For example, the vector v is

$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{A.3}$$

and the matrix m might be

$$m = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \tag{A.4}$$

Vectors are single-column matrices. Sometimes, however, subscripts are used to differentiate one vector from another. In those cases, the second element of the subscript will denote the vector component.

$$v_1(x) = \begin{bmatrix} v_{1x}(x) \\ v_{1y}(x) \\ v_{1z}(x) \end{bmatrix} \tag{A.5}$$

In this case the vector is a function of x, the x does not denote element x.

Row vectors are obtained by taking the transpose of a column vector.

$$a = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{A.6}$$

$$a^T = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix} \tag{A.7}$$

This convention differs from other books where a vector may be a row or column vector depending on the context.

A matrix may be composed of other matrices. For example, a matrix composed of three vectors is

$$\begin{bmatrix} u_x & u_y & u_z \end{bmatrix} = \begin{bmatrix} u_{xx} & u_{yx} & u_{zx} \\ u_{xy} & u_{yy} & u_{zy} \\ u_{xz} & u_{yz} & u_{zz} \end{bmatrix} \tag{A.8}$$

Each column is a vector. The first index refers to the vector and the second to the component of the vector.

A.1.2 Vector and matrix representations of operations

A few common vector operations will be described here, along with their corresponding matrix representation. In this section, vectors will be denoted with an arrow.

The dot product (or inner product) of a vector is equivalent to the transpose of a 3×1 matrix times a 3×1 matrix.

$$\vec{a} \cdot \vec{b} = a^T b = |a||b|cos\theta \tag{A.9}$$

where θ is the angle between the two vectors. The crossproduct of two vectors is equivalent to the product of the skew-symmetric form of the column matrix times the second vector.

$$\vec{a} \times \vec{b} = a^{\times} b \tag{A.10}$$

where the skew-symmetric form of a is

$$a^{\times} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{A.11}$$

Note that this particular skew-symmetric matrix is not invertible. This matrix will come into play in several areas in this book.

The last vector operation is also known as the outer product.

$$\vec{a}\vec{b} = ab^T = \begin{bmatrix} a_x b_x & a_x b_y & a_x b_z \\ a_y b_x & a_y b_y & a_y b_z \\ a_z b_x & a_z b_y & a_z b_z \end{bmatrix} \tag{A.12}$$

Throughout this book the identity matrix will be denoted by the symbol E, instead of I so that it is not confused with the rotational inertia matrix. The arrow notation denotes the quantity as a vector that is not tied to a particular coordinate frame.

A.1.3 Matrix operations

Matrix addition and subtraction are defined for matrices of the same size

$$A = B + C \tag{A.13}$$

Matrix multiplication is defined for matrices as follows

$$A(n, p) = B(n, m)C(m, p) \tag{A.14}$$

Matrix multiplication is associative

$$A(BC) = (AB)C \tag{A.15}$$

and distributive

$$A(B + C) = AB + AC \tag{A.16}$$

Generally, however

$$AB \neq BA \tag{A.17}$$

The transpose of a matrix is

$$A = B^T, A_{ij} = B_{ji} \tag{A.18}$$

and

$$(AB)^T = B^T A^T \tag{A.19}$$

A.1.4 Special matrices

A matrix is symmetric if

$$A = A^T \tag{A.20}$$

A matrix is skew-symmetric if

$$A^T = -A \tag{A.21}$$

The matrix form of the vector crossproduct is a skew-symmetric matrix, but the diagonal does not have to be zero for a matrix to be skew-symmetric.

A matrix is orthonormal if the rows and columns are orthogonal and the magnitude of each row and column is one. This is true for transformation matrices.

A.1.5 Useful matrix–vector identities

The following is a list of useful matrix–vector identities given without proof. In the following equations, C is an orthonormal matrix (each row and column is a unit vector). A and B are real matrices, and a and b are vectors (i.e., column matrices).

$$C^{-1} = C^T$$
$$-a^\times b = b^\times a$$
$$(Ca)^\times = Ca^\times C^T$$
$$-a^\times b^\times b^\times a = b^\times a^\times a^\times b$$
$$(A+B)^T = A^T + B^T$$
$$(ab)^T = b^T a^T$$
$$(a^\times)^T = -a^\times$$
$$(A^T)^{-1} = (A^{-1})^T$$
$$(AB)^{-1} = B^{-1}A^{-1}$$
$$|AB| = |A||B|$$
$$A(BD) = (AB)D$$

$$|a| = \sqrt{a^T a}$$
$$\text{Unit}(a) = \frac{a}{|a|}$$
$$\cos\theta = \frac{a^T b}{|a||b|}$$

A.2. Numerical integration

Spacecraft control-system designers rely on simulations to verify their designs prior to the construction and operation of prototypes. The term "Digital Twin" is sometimes used. Once a spacecraft is in operation, simulations are useful for studying anomalous behavior and in improving the designer's physical understanding of the device. A simulation attempts to replicate the important behavior of a system. It is not possible to represent a system exactly.

When we talk about simulation, it is convenient to break it into two categories, linear and nonlinear. Almost all real physical devices are nonlinear. Nonetheless, it is usually possible to linearize the dynamics and devices about some operating point where, in a sufficiently restricted region, the system behaves linearly. Sometimes, control designers take advantage of nonlinearities, but in many cases designs are for operating points where the system, if deviations from the operating point are small, is linear. For example, a nonlinear function is

$$y = x^3 \tag{A.22}$$

If we take the derivative at $x = x_0$ we get a linearized version

$$y = x_0^3 + 3x_0^2(x - x_0) \tag{A.23}$$

The nonlinear and linear systems coincide at $x = x_0$ as shown in Fig. A.1. The linear approximation is close to the nonlinear between $0.9 < x < 1.1$.

When dealing with the linearized system, we can work with a state-space description of the form:

$$\dot{x} = ax + bu \tag{A.24}$$
$$y = cx + du \tag{A.25}$$

where a is the plant matrix that relates the state x to the state derivative, b is the input matrix that relates the inputs, such as external disturbances and control inputs, to the state derivative, y is the measured output of the system, c relates the state to the measurements; d provides for direct feedthrough of inputs into the measurements, and u is the input. Often d is omitted, but actually it appears frequently in practice. For example,

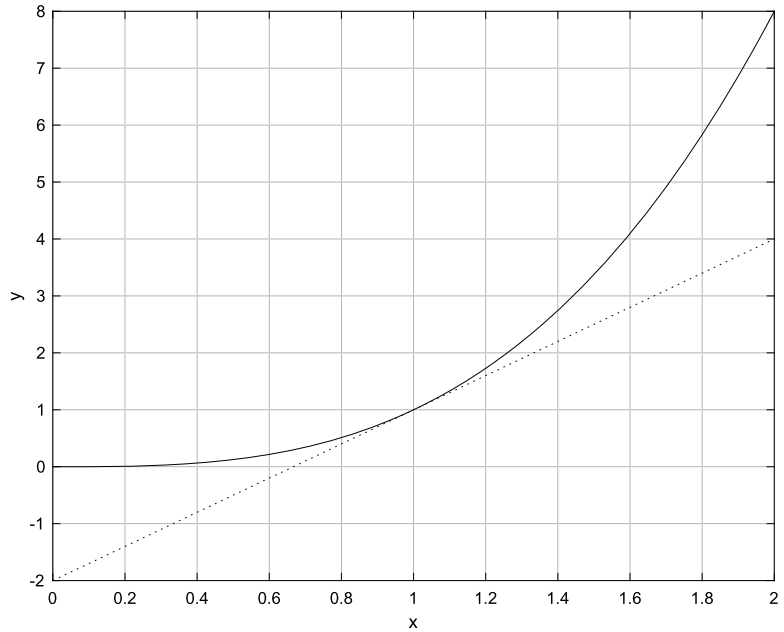


Figure A.1 Linearization of the function $y = x^3$ around $x = 1$.

the state-space description of the system

$$\dot{y} = y + \dot{\phi} \tag{A.26}$$

is

$$\dot{x} = x + \phi \tag{A.27}$$

$$y = x + \phi \tag{A.28}$$

where $d = 1$. Sometimes the state is completely measured and c is the diagonal matrix. In that case, the measurement equation is often omitted completely. Some control-design tools cannot handle systems with a nonzero d matrix.

A.2.1 Linearizing a system

Given a system of the form

$$\dot{x} = f(x, u, t) \tag{A.29}$$

$$y = g(x, u, t) \tag{A.30}$$

it can be linearized about (x_0, u_0, t_0) by expanding the function in a Taylor series to first order

$$\dot{x} = \frac{d}{dx}(f(x, u, t))_{(x_0,u_0,t_0)} + \frac{d}{du}(f(x, u, t))_{(x_0,u_0,t_0)} + f(x_0, u_0, t_0) \tag{A.31}$$

$$y = \frac{d}{dx}(g(x, u, t))_{(x_0,u_0,t_0)} + \frac{d}{du}(g(x, u, t))_{(x_0,u_0,t_0)} + g(x_0, u_0, t_0) \tag{A.32}$$

or

$$\dot{x} = ax + bu + f_0 \tag{A.33}$$

$$\delta y = cx + du + g_0 \tag{A.34}$$

If the equations are linearized about an equilibrium state, then f_0 is zero.

The simplest way to simulate a continuous-time system is to discretize it using the zero-order hold, of which there are two types. One is the standard zero-order hold. A second is the delta form of the zero-order hold. These approximations assume that the input is held constant over the interval T. This approximation is satisfactory for most spacecraft control-design problems. An alternative approach is to discretize the system using a first-order hold. This approximation assumes that the input varies linearly from step k to step $k + 1$.

Problems can arise when the dynamical system has discontinuities. For example, friction is often modeled with a Coulomb term that jumps from a negative force or torque to a positive force or torque at zero speed. If the system is linearized about this point, this effect will not be seen. Sometimes this can be solved by using a better model of the system since some discontinuities are really just first-order approximations. In the case of Coulomb friction, one can replace it with a model that has a finite slope at zero or with a bristle-friction model that has an additional friction state.

A.2.2 Nonlinear

Nonlinear systems usually require numerical integration to get a solution. This often means by Runge–Kutta integration. Assume that the system is formulated as

$$\dot{x} = f(x, u, t) \tag{A.35}$$

$$y = g(x, u, t) \tag{A.36}$$

and f is called by the integrator. All continuous models must be included in f. Discrete models, for example the digital control system, are outside of the numerical-integration routine.

Dynamics and control deals primarily with the initial-value problem.

$$\dot{x} = f(x, u, t)$$

$$\begin{aligned} y &= g(x, t) \\ x(t_0) &= x_0 \end{aligned} \tag{A.37}$$

A.2.2.1 Numerical integration methods

Assume that our problem is to solve the linear differential equation

$$\dot{x} = -x + u \tag{A.38}$$

The analytical solution is

$$x(t) = x(0)e^{-t} + (1 - e^{-t})u(t) \tag{A.39}$$

If Δt is small we might try truncating the Taylor series to first order.

$$x(t + h) = x(t) + ht(u(t) - x(t)) \tag{A.40}$$

This is known as Euler integration. By setting $t = 0$ we can compare it against the analytical solution

$$x(h) = x(0)e^{-h} + (1 - e^{-h})u(0) \tag{A.41}$$

$$x(h) = x(0)(1 - h) + hu(0) \tag{A.42}$$

which is correct since the series expansion of e^{-t} to first order is $1 - t$.

An alternative method is to compute the value of x at the time step $h/2$. This is a 2nd-order Runge–Kutta method. We write

$$x(t + h) = x(t) + hf(x(t + h/2), u(t + h/2), t + h/2) \tag{A.43}$$

$$x(t + h/2) = x(t) + \frac{h}{2} f(x(t), u(t), t) \tag{A.44}$$

The solution is

$$x(h) = x(0)(1 - h + \frac{h^2}{2}) - \frac{h^2}{2}u(0) + hu(\frac{h}{2}) \tag{A.45}$$

The approximation to the exponential is now the order of h^2. However, we need to know u at the time step $h/2$, which may be a disadvantage. This is not the only type of 2nd-order Runge–Kutta method but all involve two evaluations of f.

The standard 4th-order Runge–Kutta formula is:

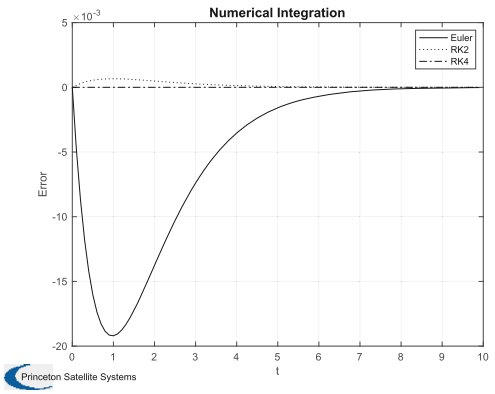
$$\begin{aligned} k_1 &= f(x, u(t), t) \\ k_2 &= f(x + \frac{h}{2}k_1, u(t + \frac{h}{2}), t + \frac{h}{2}) \end{aligned}$$

$$
\begin{aligned}
k_3 &= f(x + \frac{h}{2}k_2, u(t + \frac{h}{2}), t + \frac{h}{2}) \\
k_4 &= f(x + hk_3, u(t + h - \epsilon), t + h) \\
x &= x + \frac{h}{6}(k_1 + 2(k_2 + k_3) + k_4) + O(h^4)
\end{aligned}
\tag{A.46}
$$

The right-hand sides are computed at four different points, twice at $h/2$ and once at t and $t+h$. At $t+h$ the input term is given at $t+h-\epsilon$, where ϵ is an infinitesimal number. This means that if u is changing at every integration time step (as a digital controller might) then the previously held value is used, not the new value.

The three methods are compared in the following example, where $u = 1$ and $x(0) = 2$. All methods converge to the correct solution but the Euler method has a large initial error.

```
n  = 100;
p  = zeros(3,n);
x1 = 2; x2 = 2; x3 = 2;
h  = 0.1;
for k = 1:n
  p(:,k) = [x1;x2;x3];
  x1     = Euler( @RHSLinear, x1, h, 0 );
  x2     =   RK2( @RHSLinear, x2, h, 0 );
  x3     =   RK4( @RHSLinear, x3, h, 0 );
end
t = (0:(n-1))*h;
e = 2*exp(-t) + 1-exp(-t);
Plot2D(t,p - [e;e;e],'t','Error','Numerical␣
      Integration');
legend('Euler', 'RK2', 'RK4');

%% Right hand side
function xDot = RHSLinear( x, t )

xDot = 1 - x;

end

function x = Euler(Fun,x,dT,t,varargin)

x = x + dT*feval(Fun,x,t,varargin{:});

end
```



Example A.1: Numerical integration.

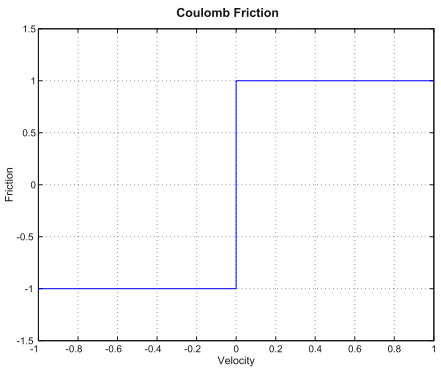
This method does not provide an error estimate. However, it is possible to run a 4th- and 5th-order (or in general n- and $n+1$-order) method with only one more calculation of f. The difference between the solutions can be used to adjust the step size. Since the step size is a scalar some scalar comparison must be made. For spacecraft problems, angular momentum is the best quantity to use for this purpose. In general, any quantity that is conserved is a good choice for choosing the step size. General-purpose solvers do not know anything about the problem so they use the entire state for error correction.

Some solvers can also vary the order of the method. With Runge–Kutta this can be done every step since a step is entirely self-contained, i.e., step k does not rely on step $k-1$.

A.2.3 Discontinuities

Many mechanical systems have nonlinearities that are modeled using discontinuous functions. Coulomb friction is an example. Coulomb friction is shown in Example A.2.

```
1 Plot2D([-1 0 0 1],[-1 -1 1 1],'Velocity','
      Friction', 'Coulomb Friction')
2 set(gca,'ylim',[-1.5 1.5]);
```



Example A.2: Coulomb friction.

When numerically integrating a function with Coulomb friction, the solution will limit cycle, that is manifest a self-sustained oscillation. This would imply that a mechanism with Coulomb friction would oscillate forever. Rather than using this kind of function, the discontinuous function can be replaced with a smooth function. For example, the function

$$y = \frac{2}{1 + e^{-bx}} - 1.0 \tag{A.47}$$

produces Example A.3. If the device never passes through zero then the discontinuous model is fine. If it does, you may need another numerical model.

A.3. Fourier series

Fourier series are a useful way to study the behavior of a system as a function of frequency. This section will describe Fourier series using trigonometric functions.

A.3.1 Trigonometric identities

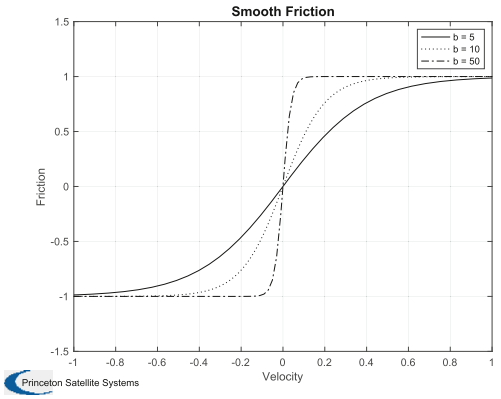
Trigonometric sum of angle formulas and sum of squares formulas.

$$\sin(a \pm b) = \sin a \cos b \mp \sin b \cos a \tag{A.48}$$

$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b \tag{A.49}$$

$$1 = \sin^2 a + \cos^2 a \tag{A.50}$$

```
x = linspace(-1,1);
y0 = 2./(1 + exp(-5*x)) - 1;
y1 = 2./(1 + exp(-10*x)) - 1;
y2 = 2./(1 + exp(-50*x)) - 1;
Plot2D(x,[y0;y1;y2],'Velocity','Friction', '
      Smooth Friction');
legend('b = 5','b = 10', 'b = 50');
set(gca,'ylim',[-1.5  1.5]);
```



Example A.3: Smooth friction.

You can get many useful relationships from these. For example:

$$\cos(2a) = \cos^2 a - \sin^2 a = \cos^2 a - (1 - \cos^2 a) = 2\cos^2 a - 1 \tag{A.51}$$

$$\cos^2 a = \frac{1}{2}(1 + \cos(2a)) \tag{A.52}$$

A.3.2 Sine and cosine Fourier series

If the function is periodic over $-\pi$ to π.

$$f(x) = a_0 + \sum_{k=1}^{\infty} \left(a_k \cos nx + b_k \sin nx\right) \tag{A.53}$$

where

$$a_0 = \frac{2}{\pi} \int_{-\pi}^{\pi} f(x)\ dx \tag{A.54}$$

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx dx \tag{A.55}$$

$$b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx dx \tag{A.56}$$

If the function is periodic over the interval $-L$ to L let $t = \frac{Lx}{\pi}$. The Fourier series is now

$$f(x) = a_0 + \sum_{k=1}^{\infty} \left(a_k \cos\left(n\frac{\pi t}{L}\right) + b_k \sin\left(n\frac{\pi t}{L}\right)\right) \tag{A.57}$$

where

$$a_0 = \frac{2}{L}\int_{-L}^{L} f(x)\ dx \tag{A.58}$$

$$a_k = \frac{1}{L}\int_{-L}^{L} f(x)\cos\left(n\frac{\pi x}{L}\right) dx \tag{A.59}$$

$$b_k = \frac{1}{L}\int_{-L}^{L} f(x)\sin\left(n\frac{\pi x}{L}\right) dx \tag{A.60}$$

A.4. Spherical geometry

Spherical geometry is an alternative way of looking at unit vectors. Fig. A.2 shows the unit sphere and three unit vectors. There are two angles that are important in spherical

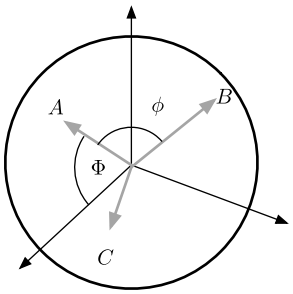


Figure A.2 Unit sphere and unit vectors.

geometry. The first is the angle ϕ between the vectors, which is found from the dot product,

$$A^T B = \cos(\phi) \tag{A.61}$$

which is the dot product. The second angle is the angle Φ between the arcs, for example angle Φ between arcs AC and AB. Spherical trigonometry is useful when using vectors to stars or planets for measurements. Two of the most important equations are the law of sines and law of cosines for spherical angles. The law of sines is

$$\frac{\sin(\theta)}{\sin(\Theta)} = \frac{\sin(\lambda)}{\sin(\Lambda)} = \frac{\sin(\phi)}{\sin(\Phi)} \tag{A.62}$$

The law of cosines for the sides is

$$\cos(\lambda) = \cos(\theta)\cos(\phi) + \sin(\theta)\sin(\phi)\cos(\Lambda) \tag{A.63}$$

and the law of cosines for the angles is

$$\cos(\Lambda) = \cos(\Theta)\cos(\Phi) + \sin(\Theta)\sin(\Phi)\cos(\lambda) \tag{A.64}$$

The small Greek letter denotes the angle from the dot product. The capital Greek letter denotes the angle between the arcs on the surface of the sphere that is opposite the corresponding small letter. The three arcs form a spherical triangle.

A.5. The chain rule in calculus

The chain rule often appears in derivations

$$\frac{dab}{dt} = a\frac{db}{dt} + \frac{da}{dt}b \tag{A.65}$$

Sometimes we want to change the independent variable in a differential equation. For example,

$$\frac{dx}{dt} + \sigma x = u \tag{A.66}$$

Suppose we want to differentiate with respect to θ and $\theta = \omega t$. If ω is a constant then

$$d\theta = \omega dt \tag{A.67}$$

or

$$\frac{dx}{d\theta} + \sigma\frac{x}{\omega} = \frac{u}{\omega} \tag{A.68}$$

APPENDIX B

Probability and statistics

B.1. Space story

I met Professor Van der Velde at MIT to talk about estimation. I was thinking of taking his estimation course. He told me that he always gave out a list of 100 probability problems for students to do as a review for the course as it was assumed that graduate students had some background in the material. He said that each year, without fail, the students could hardly do any of the problems despite having taken courses in probability and statistics.

B.2. Introduction

Probability and statistics are a critical aspect of attitude control and determination systems. They enter into the discipline in many ways. For example,

1. Sensor measurements have noise.
2. Certain disturbances on a spacecraft are best characterized as noise.
3. Alignments of sensors and actuators on the spacecraft are only known with a degree of uncertainty. This must be considered in pointing budgets.
4. Actuators, sensors, and supporting hardware have a probability of failure and an expected lifetime.
5. When landing on a planet, moon, or asteroid, the surface is not known perfectly. There is some probability that the landing spot will have a boulder or be too soft to support the lander.
6. For a debris-removal satellite, the distribution of debris is not known perfectly and the angular rates of the debris may not be known.
7. Link budgets, which provide requirements, are impacted by the signal-to-noise ratio of the telecommunications systems. Noise is a random process.

This appendix discusses concepts in probability and statistics that are needed for aircraft control-system design. In particular, it discusses stochastic processes that are used for modeling gusts and other phenomena, and the characterization of noise models for sensors. Stochastic processes appear in several areas of aircraft control:

1. External disturbances: wind and wind gusts;
2. Noise: sensor noise;
3. Tolerances: mechanical tolerances on sensor alignments, actuator alignments, actuator and sensor scale factors, etc.

B.3. Axiomatic probability

The basic notion of probability is

$$P(A) = \frac{\text{Number Of Outcomes With A}}{\text{Total Outcomes}} \tag{B.1}$$

For a coin flip there are two outcomes, heads or tails and one outcome favoring heads therefore the probability of heads is 0.5. This definition assumes that all outcomes are equally likely. A more general method of approaching probability is an axiomatic probability that deals with sample spaces. A sample space is the set of all possible outcomes. Each outcome is called an element or point in the sample space. The number of points in the sample space may be finite, countably infinite or infinite. For example, in the experiment of drawing from a deck of cards, there are 52 possible outcomes. Thus the sample space consists of a list of all of the cards in the deck. The axioms of probability are that the probability of any outcome is greater than or equal to zero. The probability that at least one outcome will occur is 1 and if the events are mutually exclusive the probability of any set of the events occurring is equal to the sum of the probabilities of each event occurring. In equation form, the probabilities are

$$P(A) \geq 0 \tag{B.2}$$

$$P(S) = 1 \tag{B.3}$$

$$P(A_1 \cup A_2 \cup A_3) = P(A_1) + P(A_2) + P(A_3) \tag{B.4}$$

where $\cup$ is the union symbol and means combining the sets. Sometimes you want to know the probability of one outcome given that another has already occurred. This is the conditional probability and is given as

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{B.5}$$

where $\cap$ is the intersection of the two sets. This leads to Bayes rule, which is

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{B.6}$$

Events A and B are independent if

$$P(A \cap B) = P(A)P(B) \tag{B.7}$$

If this is the case the conditional probability is

$$P(A|B) = \frac{P(B)P(A)}{P(B)} = P(A) \tag{B.8}$$

B.4. Binomial theorem

The binomial theorem

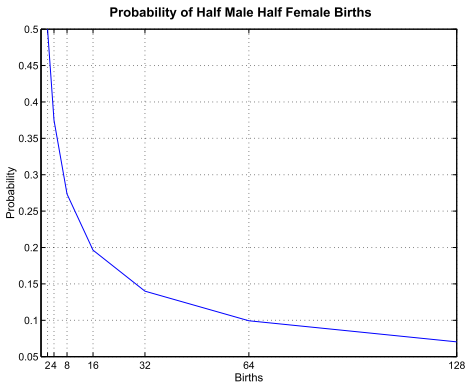
$$p(k|n) = \frac{n!}{(n-k)!k!} p^k (1-p)^{n-k} \tag{B.9}$$

gives the probability of k occurrences in n samples if the probability of one occurrence is p. For example suppose two dogs are born. What is the probability that one is female and one male given that the probability of a male is 0.5? This can be done by simple counting. The four possibilities are (where 0 is male and 1 is female):

00
01
10
11

There are four possibilities of which two have one male and one female. Thus the probability is 0.5. However, for larger numbers of samples the probability of there being half males and half males decreases because the number of cases in which there are different numbers of males and females grows in relation to the number of cases where there are half males and half females. See Example B.1.

```
births = [2 4 8 16 32 64 128];

pC = zeros(1,7);
for j = 1:length(births)
  pC(j) = BinomialTheorem( births(j), births(j)
          /2, 0.5 );
end

Plot2D( births, pC, 'Births','Probability',...
 'Probability of Half Male Half Female Births')
set( gca, 'xlim', [0 births(end)], 'xtick',
        births )
```



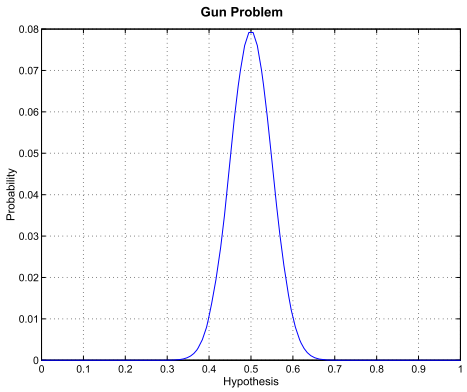
Example B.1: Dog-birth problem.

The binomial theorem can be used for hypothesis testing. Suppose that a gun fires 100 shots and hits 50 times. What is the probability that the gun is 50% accurate? We write

$$p(k|n) = \frac{100!}{50!50!} 0.5^k (0.5)^{n-k} \tag{B.10}$$

which is about 8%. ! is the factorial symbol $3! = 3 \times 2 \times 1$. This seems low, but Example B.2 shows that this is the most likely hypothesis. Thus the probability of the hypothesis being true must be compared to the probability of any other hypothesis being true.

```
p = linspace(0,1);

pC = zeros(1,length(p));
for j = 1:length(p)
  pC(j) = BinomialTheorem( 100, 50, p(j) );
end

Plot2D( p, pC, 'Hypothesis', 'Probability', 'Gun␣
      Problem')
```



Example B.2: Hypothesis testing.

B.5. Probability distributions

The cumulative probability-distribution function is written as

$$F(x) = P[X \leq x] \tag{B.11}$$

$$F(\infty) = 0 \tag{B.12}$$

This says that the probability that X is between $\pm\infty$ is 1. $F(x)$ is the probability that the value of the random variable X is less than x. The probability-density function (pdf) is the derivative of the cumulative probability-distribution function or

$$f(x) = \frac{d}{dx}F(x) \tag{B.13}$$

and

$$F(x) = \int_{-\infty}^{x} f(x)dx) \tag{B.14}$$

If the integral equals 1, then the probability that X is between two values a and b is

$$P[a < X \leq b] = F(b) - F(a) \tag{B.15}$$

$$= \int_{-\infty}^{b} f(x)dx - \int_{-\infty}^{a} f(x)dx = \int_{a}^{b} f(x)dx \tag{B.16}$$

The mean, or expected value E, is

$$E(X) = \mu = \int_{-\infty}^{\infty} xf(x)dx \tag{B.17}$$

The root mean-squared value is

$$E(X^2) = \int_{-\infty}^{\infty} x^2 f(x)\,dx \tag{B.18}$$

The variance is

$$Var(X) = \sigma^2 = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)\,dx \tag{B.19}$$

so that

$$\sigma = E(X^2) - \mu^2 \tag{B.20}$$

The variance of a random variable multiplied by a scalar can be easily found and is

$$Var(aX) = a^2\, Var(X) \tag{B.21}$$

If random variables are independent, that is the value of one does not depend on the value of another (such as the flipping of two points), the mean value of the product is

$$E(X_1 X_2) = E(X_1)E(X_2) \tag{B.22}$$

and the variance of the sum is

$$Var(X_1 + X_2) = Var(X_1) + Var(X_2) \tag{B.23}$$

An estimate of the mean value of X can be found by taking N samples

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{B.24}$$

The sample variance is

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N} (X_i - \bar{X})^2 \tag{B.25}$$

Note that if instead of $N-1$ in the denominator we had N we would be computing the population variance. The sample standard deviation is

$$S = \sqrt{S^2} \tag{B.26}$$

Note that each value in the sum is in itself a random variable since each measured value is subject to chance. Thus

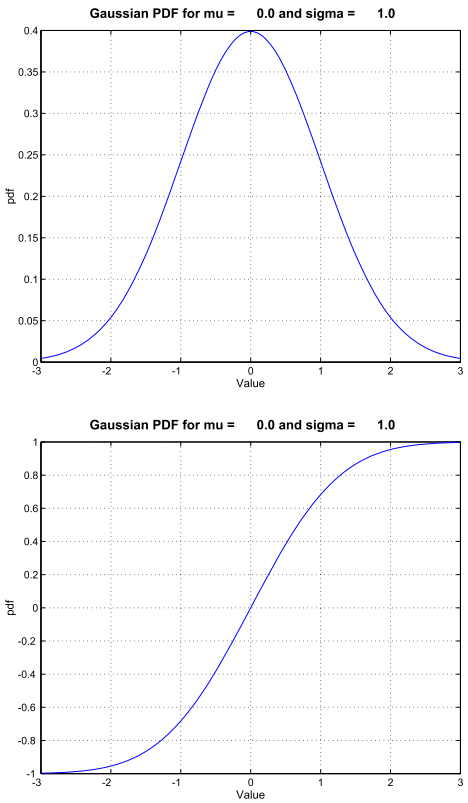
$$E(\bar{X}) = E\left(\frac{1}{N}\sum_{i=1}^{N} X_i\right) = \frac{1}{N}\sum_{i=1}^{N} E(X_i) = \frac{N\mu}{N} \tag{B.27}$$

Thus the sample mean is an unbiased estimate of the mean value m. Likewise $E(S^2) = \sigma^2$. In the limit, as you sum a large number of independent random variables you get a Gaussian or normal distribution. The probability density function for the Gaussian distribution is

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{B.28}$$

With a Gaussian distribution 68% of the values will fall within $\pm\sigma$, 95% of the values will fall within $\pm 2\sigma$, and 99.7% of the values will fall within $\pm 3\sigma$ The Gaussian pdf and cpdf are shown in Example B.3. Both figures are over the same 3 σ range. By the way, 6 σ is 99.9999998.

```
x     = linspace(-3,3);
mu    = 0;
sigma = 1;
pDF   = GaussianPDF( x, mu, sigma );
cPDF  = GaussianCPDF( x, mu, sigma );
t     = sprintf('Gaussian PDF for mu = %8.1f and sigma = %8.1f',mu,sigma);
Plot2D( x, pDF, 'Value', 'pdf', t );
Plot2D( x, cPDF, 'Value', 'pdf', t );
```



Example B.3: Gaussian pdf and cpdf.

B.6. Evaluating measurements

Two measures for evaluating measurements are the confidence interval and sign test. The confidence interval is based on the Gaussian pdf and tells you the probability that a value will lie with $x\%$ of the nominal. A way of comparing two identical instruments is the sign test. Given two sequences of measurements done under identical circumstances we subtract one set from the other and count positive signs. If the errors are truly random then half the signs will be positive and the other half negative.

B.7. Combining errors

If a quantity can be expressed in the form

$$y = f(x_1, x_2) \tag{B.29}$$

then the contribution of errors in x_1 and x_2 to y can be found using the following formulas. Let

$$a = \left[\frac{\partial f}{\partial x_1} \mid_x \frac{\partial f}{\partial x_2} \mid_x \right]^T \tag{B.30}$$

where x is the current or nominal value of the state vector. Then, the error estimate is

$$z = \frac{xa}{y} \tag{B.31}$$

$$e_{abs} = \sum_x |z_k| e_k \tag{B.32}$$

$$e_{SRSS} = \sqrt{\sum_k (z_k e_k)^2} \tag{B.33}$$

where e is the fractional error for each element of x. xa is

$$xa = \left[\begin{array}{cccc} a_1 x_1 & a_2 x_2 & \ldots & a_N x_N \end{array} \right] \tag{B.34}$$

The first error is more conservative than the second.

B.8. Multivariate normal distributions

The multivariate normal distribution is important in many areas of control. It is defined as

$$f(x) = \frac{1}{(s\pi)^{n/2}\sqrt{|C|}} e^{-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)} \tag{B.35}$$

where C is the covariance matrix and is

$$C = \begin{bmatrix} E[(X_1 - \mu_1)^2] & E[(X_1 - \mu_1)(X_2 - \mu_2)] \\ E[(X_1 - \mu_1)(X_2 - \mu_2)] & E[(X_2 - \mu_2)^2] \end{bmatrix} \tag{B.36}$$

$$C = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix} \tag{B.37}$$

for a two-dimensional system. σ is the variance and ρ the correlation coefficient. The determinant in this case is

$$|C| = (1 - \rho^2)\sigma_1^2\sigma_2^2 \tag{B.38}$$

which is zero if $\rho = 1$ because the two variables are completely correlated, i.e., indistinguishable and not independent.

B.9. Random signals

A random signal is any signal that is not deterministic. The following is a deterministic signal

$$x(t) = \sin(t) \tag{B.39}$$

where $x(t)$ is exactly determined as a function of t. A random signal might be one in which the amplitude is random

$$x(t) = A\sin(t) \tag{B.40}$$

This would be the equivalent of having a sine-wave generator in a lab in that the amplitude of the wave was determined by a dial that is set to the value that the last engineer used. When you turn on the generator you would get a sine wave with a constant amplitude but the value of that amplitude would be random. There are two important definitions. Assume that you run an experiment in which you measure a signal. You then run the same experiment many times. This produces an ensemble of time signals. If averaging over a single time signal is the same as averaging over the ensemble of time signals then the process is ergodic. If the probability density functions do not change as a function of time, the process is time stationary. The autocorrelation function for a random process $X(t)$ is

$$R_X(T_{1,t_2}) = E[X(t_1)X(t_2)] \tag{B.41}$$

If the process is stationary

$$R_X(\tau) = E[X(t)X(t+\tau)] \tag{B.42}$$

The crosscorrelation is written as

$$R_{XY}(\tau) = E[X(t)Y(t+\tau)] \tag{B.43}$$

for a stationary process. The power spectral density is the Fourier transform of the crosscorrelation and is

$$S_X(j\omega) = \int_{-\infty}^{\infty} R_X(\tau) e^{-j\omega\tau} d\tau \tag{B.44}$$

The cross-spectral density is defined in a similar fashion. White noise is defined as a stationary random process having a constant spectral density function. Thus

$$S_{X(j\omega)} = A \tag{B.45}$$

and the crosscorrelation is

$$R_X(\tau) = A\delta(\tau) \tag{B.46}$$

where $\delta(\tau)$ is the unit impulse, meaning that the value at any one time is unrelated to the value at any other time.

B.10. Outliers

If we assume that the noise-corrupting data point is Gaussian, the probability that a noisy point lies within a distance of its true point decreases rapidly with the distance. For the Gaussian hypothesis to be correct, noise should not move points more than a few standard deviations. If a small percentage of points is greater than 2 or 3 standard deviations then the resulting estimate will be poor since the data is not conforming to the hypothesis on which the least-squares estimator is based. For this reason, outliers should be eliminated from the data sample.

B.11. Noise models

Random variables are needed when simulating spacecraft. They are used for sensor-noise models and for Monte Carlo simulations. Monte Carlo simulations are when you run a simulation numerous times with randomly distributed initial conditions and randomly distributed parameters. Uniform random numbers and normal random numbers are easily obtained. Some noise models require other probability distributions. A general and simple technique is the inverse cumulative function (CDF) sampling technique [1]. A continuous CDF is a one-to-one mapping of the domain of the CDF into the interval from zero to one. Knowing the inverse CDF allows us to go backwards from a random-number generator that produces a uniform distribution between 0 and 1 and

any other distribution. For example, the CDF of an exponential distribution is

$$f(t) = \theta e^{-\theta t} \tag{B.47}$$

The inverse CDF is

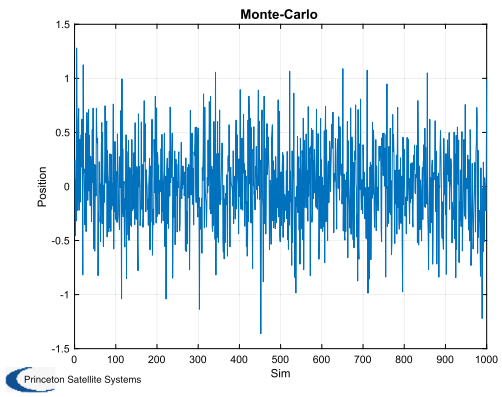
$$F^{-1}(u) = -\frac{\log(1-u)}{\theta} \tag{B.48}$$

where u is a random number between zero and 1. θ is the scaling.

B.12. Monte Carlo methods

Monte Carlo methods, (named after the famous casino) are computational algorithms that rely on random sampling to obtain results. For example, a Monte Carlo approach to a damped second-order system simulation would be to use randomly generated step inputs to determine the performance of the system. Example B.4 shows a Monte Carlo simulation of a damped second-order system in which the input is a Gaussian random number and the damping ratio is a uniform random number between 0 and 1. This might be used for a slosh study where the damping is not well known.

```
%% Monte-Carlo simulation

n  = 1000;
xP = zeros(1,n);
dT = 0.1;

for j = 1:n
  x    = [0;0];
  a    = 0.01*randn;
  zeta = rand;
  for k = 1:100
    x = RK4(@RHS,x,dT,0,a,zeta);
  end
  xP(1,j) = x(1);
end

Plot2D(1:n,xP,'Sim','Position','Monte-Carlo');


function xDot = RHS(x,~,a,zeta)

omega = 0.1;

xDot = [x(2);-zeta*omega*x(2) - omega^2*x(1) + a
       ];

end
```



Example B.4: Monte Carlo simulation with a Gaussian random input and a uniform random-number damping ratio.

References

[1] R. Wicklin, Simulating Data with SAS, SAS Institute, April 2013.

APPENDIX C

Time

This chapter introduces time scales that are used in astronomical and spacecraft work. Time is needed to determine the orientation of a planet with respect to the inertial frame. It is also needed for star cameras to accurately determine the azimuth and elevation of stars in the star field. For deep-space spacecraft, time is needed to determine the positions of the planets relative to the spacecraft.

C.1. Time scales

A time scale is defined by a specific periodic natural phenomenon. The formal definition must include a description of the phenomenon to be used (i.e., what defines a period), the rate of advance (the ratio of time units to the natural period) and an initial epoch, which is a time reading of some identifiable event.

There are two time scales used in astronomy. The first is based on the "atomic" second in Systeme International (SI) with an SI second corresponding to 9 192 631 770 cycles of the radiation corresponding to the ground-state hyperfine transition of Cesium 133. The second is based on the rotation of the Earth. Since the rotation of the Earth is not constant, the relationship between the two time scales is complex.

Important time scales are:

- UT1 (Universal Time 1)—A time scale based on the rotation of the Earth. UT1 is defined with respect to Sidereal Time.
- TT (Terrestrial Time)—The theoretically ideal time scale at the surface of the Earth.
- TDT (Terrestrial Dynamical Time)—A time scale that would be kept by an ideal clock at sea level.
- TDB (Barycentric Dynamical Time)—A time scale that would be kept by an ideal clock moving with the solar-system barycenter. It is always within 2 milliseconds of TDT and the difference is caused by relativistic effects.
- TAI (International Atomic Time)—A time scale kept by atomic clocks on the Earth.
- UTC (Universal Time Coordinated)—The basis for international time keeping. Its rate is the same as TAI, but its epoch is adjusted in one-second steps to keep it within 0.9 s of UT1, which is measured by the Earth's rotation against the stars.
- DUT1 (Universal Time Correction)—The difference between UT1 and UTC.
- Sidereal Time—Measured by the rotation of the Earth against the stars.

The standard epoch for modern astrometric reference data, designated J2000.0, is expressed as a Terrestrial Time TT instant: J2000.0 is 2000 January 1, 12h TT (JD 2451545.0 TT) at the geocenter.

Universal time (UT) now always refers to UT1. UT1 is a linear function of the Earth's rotation angle θ, which is the geocentric angle between two directions in the equatorial plane called the Celestial Intermediate Origin (CIO) and the Terrestrial Intermediate Origin (TIO). The TIO rotates with the Earth but CIO does not, so that $\dot{\theta} = \omega$, the Earth's angular rate. However, ω is not a constant and must be measured through astronomical observations.

The worldwide system of civil time is based on Coordinated Universal Time (UTC). UTC uses the SI second on the geoid as its fundamental unit, but is subject to occasional 1-second adjustments to keep it within 0.9 second of UT1. Such adjustments, called leap seconds, are normally introduced at the end of June or December, when necessary, by international agreement.

$$TAI = UTC + \Delta AT \tag{C.1}$$

where ΔAT is the accumulated number of leap seconds. The astronomical time scale Terrestrial Time is

$$TT = TAI + 32.184s \tag{C.2}$$

Geocentric coordinate time is

$$TCG = \frac{TT - L_G t_0}{1 - L_G} \tag{C.3}$$

where $L_G = 6.969290134 \times 10^{-10}$.

TT, *TCG*, and *TCB* all read 1977 January 1 00:00:32.184 (JD 2443144.5003725) on 1977 January 1, 00:00:00 TAI (JD 2443144.5 TAI) at the geocenter. This event is t_0 in the above equations.

The relationship between *TDB* and *TT* is more complex but can be approximated by the series

$$\begin{aligned} TDB \approx TT &+ 0.001657 \sin(628.3076T + 6.2401) \\ &+ 0.000022 \sin(575.3385T + 4.2970) \\ &+ 0.000014 \sin(1256.6152T + 6.1969) \\ &+ 0.000005 \sin(606.9777T + 4.0212) \\ &+ 0.000005 \sin(52.9691T + 0.4444) \\ &+ 0.000002 \sin(21.3299T + 5.5431) \\ &+ 0.000010 \sin(628.3076T + 4.2490) + \ldots \end{aligned} \tag{C.4}$$

where T is the number of Julian centuries from J2000.0 or

$$T = \frac{JD(TT) - 2451545.0}{36525} \tag{C.5}$$

The maximum error using the above expansion is 10 μs between the years 1600 and 2200, which is two orders of magnitude better than $TT = TDB$.

$$UT1 = UTC + (UT1 - UTC) \tag{C.6}$$

or

$$UT1 = UTC + DUT1 \tag{C.7}$$

where DUT1 is a broadcast approximation to UT1-UTC, which is accurate to 0.1 second,

$$UT1 = UTC - \Delta T \tag{C.8}$$

where

$$\Delta T 32.184 + \Delta AT - (UT1 - UTC) \tag{C.9}$$

Values of ΔT are listed in the Astronomical Almanac [1].

C.2. Earth rotation

The Earth rotates about its axis. This rotation is measured with respect to the inertial frame. The terrestrial day is referenced to Earth rotation.

The Earth's rotation angle, θ, is

$$\theta = 0.7790572732640 + 1.00273781191135448 D_U \tag{C.10}$$

where

$$D_U = JD(UT1) - 2451545.0. \tag{C.11}$$

A more accurate formula is

$$\theta = 0.7790572732640 + 0.00273781191135448 D_U + \text{frac}(JD(UT1)) \tag{C.12}$$

Greenwich Mean Time (GMT) is

$$GMST = 86400\,\theta + (0.014506 + 4612.156534\ T + 1.3915817\ T^2 - 0.00000044\ T^3 - 0.000029956\ T^4 - 0.0000000368\ T^5)/15 \tag{C.13}$$

where T is the number of centuries in TDB (or TT). Sidereal time is the time used by astronomers to locate celestial objects. The apparent sidereal time is

$$GAST = GMST + \frac{\epsilon_T}{15} \tag{C.14}$$

where ϵ_T is the Equation of the Equinoxes, which is approximately

$$E = \psi \cos\epsilon \tag{C.15}$$
$$+ 0.00264096 \sin(\Omega) \tag{C.16}$$
$$+ 0.00006352 \sin(2\Omega)$$
$$+ 0.00001175 \sin(2F - 2D + 3\Omega)$$
$$+ 0.00001121 \sin(2F - 2D + \Omega)$$
$$- 0.00000455 \sin(2F - 2D + 2\Omega)$$
$$+ 0.00000202 \sin(2F + 3\Omega)$$
$$+ 0.00000198 \sin(2F + \Omega)$$
$$- 0.00000172 \sin(3\Omega)$$
$$- 0.00000087 T \sin(\Omega) + ...$$

where ψ is the nutation in longitude, in arcseconds; ϵ is the mean obliquity of the ecliptic; and F and D, are fundamental lunisolar arguments. Mean sidereal time does not use the Equation of the Equinoxes.

Local sidereal time is

$$LXST = GXST + \frac{3600}{15}\lambda \tag{C.17}$$

where X is A or M and λ is the longitude of the place of interest in degrees.

C.3. Julian date

Julian date gives a continuous count of dates since January 1, 4713 BC at noon. Integer Julian Dates always refer to noon. The Julian date for the standard epoch J2000.0, January 1, 2000 12h is JD 2451545.0 TDB.

Julian date is very convenient in control work because it does not require any complex logic, as would years and days. However, to get high accuracy you need a lot of digits. Accuracy to the day requires 7 digits. Accuracy to the millisecond requires another 9 digits. Sometimes Modified Julian Date (MJD) is used instead. The MJD begins at midnight rather than noon.

$$MJD = JD - 2400000.5 \tag{C.18}$$

With double-precision numerics on modern-day processors, MJD is not very important.

Knowledge of the Julian date is also required when transforming between Earth-Centered Inertial (ECI) and Earth-Fixed (EF) coordinate frames. Earth-Fixed coordinate frames rotate with the Earth, and the Julian date is required in order to compute the amount of rotation that has occurred since the last reference epoch.

Another important time is the Greenwich Mean Sidereal Time, which is the angle between the ECI *x*-axis (Vernal Equinox) and the Greenwich Meridian.

C.4. Time standards

The following sections discuss terrestrial time standards. If you have a vehicle on another planet then your day will be determined by the rotation of that planet.

C.4.1 Local time

The date/time that you see on local clocks or on the Internet.

C.4.2 UTC

Coordinated Universal Time, more precisely known as GMT (Greenwich Mean Time), or Zulu time. Local times around the world differ by time zones. For example, Eastern Standard Time is exactly one hour later than Atlantic time. Some time zones have standard and daylight times. There are currently 37 different time zones in use around the world.

C.4.3 GPS

Global Positioning System time, is the time scale implemented by the atomic clocks in the GPS ground-control stations and the GPS satellites. GPS time was zero at midnight January 6, 1980.

Table C.1 Planetary days.

Celestial Object	Day (Earth Hours)
Mercury	1408
Venus	5832
Earth	24
Mars	25
Jupiter	10
Saturn	11
Uranus	17
Neptune	16
Pluto	153
Europa	84 (tidally locked to its orbit period)
Enceladus	32.9 (tidally locked to its orbit period
Titan	382 (tidally locked to its orbit period)
Moon	672 (tidally locked to its orbit period)

C.4.4 Loran-C

Long-Range Navigation time, is a time scale measured by the atomic clocks in Loran-C chain transmitter sites. Loran time was zero at midnight January 1, 1958.

C.4.5 TAI

Temps Atomique International, is the international atomic time scale based on a continuous counting of the International System of Units second. TAI leads GPS time by 19 seconds.

C.4.6 Planetary days

Table C.1 lists planetary day lengths for selected solar-system objects.

References

[1] P.K. Seidelmann, Explanatory Supplement to the Astronomical Almanac, University Science Books, Mill Valley, CA, 1992.

APPENDIX D

Coordinate systems

A right-handed reference frame is defined by an origin and a set of three orthogonal axes in a right-handed set. The origin is often either the center of the Earth or the center-of-mass of the vehicle. In general, however, the origin of the body frame of a vehicle is defined at a selected point, and the center-of-mass varies as fuel is consumed or articulated bodies, such as robot arms, are moved. The point is often chosen during manufacturing as a reference.

D.1. Earth-centered inertial coordinates

The inertial reference frame used is the J2000.0 frame. J2000.0 defines the date January 1, 2000 at 12h Universal Time. This frame has its xy-plane parallel to the mean Earth equator at epoch J2000.0 and its z-axis pointing towards the mean North celestial pole of J2000.0. The x-axis points toward the mean vernal equinox of J2000.0. Greenwich is the meridian that passes through the Royal Observatory in Greenwich in the United Kingdom. This frame is also sometimes called the Earth-Centered Inertial frame (ECI). It is shown in Fig. D.1 with the Earth-Fixed (EF) and local vertical/local horizontal (LVLH) frames.

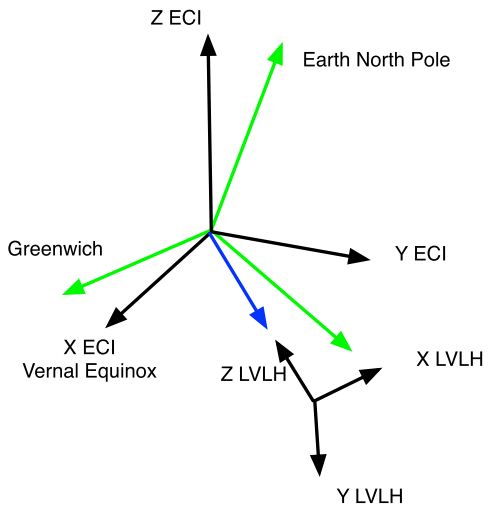


Figure D.1 Coordinate frames.

The black-axis system is the ECI system. The green is EF. The z-axis for the EF system is along the true North pole of the Earth and the x-axis is along the Greenwich meridian. The z-axis for the ECI frame is along the mean North pole of the Earth. The EF system rotates about the z-axis once per day. The deviation between the true North pole and mean North pole is very small and can usually be neglected.

The transformation matrix between the ECI frame and the Earth-Fixed frame is the product of three matrices, G, N, and P

$$M = GNP \tag{D.1}$$

where G is the Earth rotation about the pole, N is the Earth nutation, and P is the Earth precession.

D.2. Local vertical/local horizontal coordinates

The LVLH (local vertical/local horizontal) frame is a rotating frame and is very convenient for Earth-oriented spacecraft. It has the $+z$-axis pointing at the Earth (nadir), $-y$ along the orbit normal, and $+x$ completing the right-hand set. This frame rotates at orbit rate and is the same as the Earth-centered frame when the vehicle is in an equatorial orbit. "Mean" coordinates for the Earth neglect the Earth's nutational motion. If Earth nutation is included, the coordinates are known as true.

A few other definitions are in order. Nadir refers to the vector from the center of the vehicle to the center of the Earth. Zenith points away from the Earth. In the Earth-pointing frame nadir is the $+z$ axis and zenith is $-z$. The nadir vector would always be parallel to the vehicle position vector if the Earth were spherically symmetric, see Fig. D.2.

The LVLH matrix (that transforms from ECI to LVLH) is

$$m = \begin{bmatrix} x y z \end{bmatrix} \tag{D.2}$$

where

$$y = \frac{v^{\times} r}{|v^{\times} r|} \tag{D.3}$$

$$z = -\frac{r}{|r|} \tag{D.4}$$

$$x = \frac{y^{\times} z}{|y^{\times} z|} \tag{D.5}$$

where x, y, and z are column vectors that form the matrix. Converting between frames involves computing the rotation matrices that perform the transformation between the frames. Celestial objects are often referred to in the Earth-centered frame.

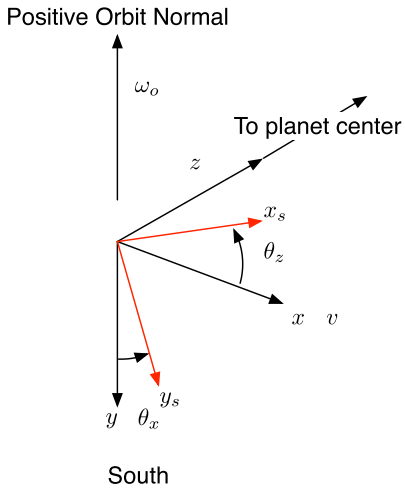


Figure D.2 Local vertical/local horizontal (LVLH) frame.

D.3. Heliocentric coordinates

The heliocentric frame z-axis is normal to the ecliptic plane, that is the plane of the Earth's orbit. The x-axis is along the line of Aries that points through the Earth to the constellation of Aries. The transformation matrix from ecliptic to the Earth equatorial planes is a single rotation about the x-axis.

$$T = (J - 2\,451\,545)/36\,525 \tag{D.6}$$

$$\theta = ((23.43929111 - ((46.815 + (0.00059 - 0.001813T)T)T)/3600) \tag{D.7}$$

where J is the Julian date in days and T is Julian centuries. The transformation matrix is

$$c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{D.8}$$

D.4. International Space Station coordinates

Many spacecraft need to operate near the International Space Station (ISS).

The ISS frame is defined as:

1. $+X$ Along the velocity vector (Forward);
2. $-X$ Opposite the velocity vector (Aft);
3. $+Y$ Negative Orbit Normal (Starboard);

4. $-Y$ Positive Orbit Normal (Port);
5. $+Z$ Zenith (Deck);
6. $-Z$ Nadir (Overhead).

The coordinates are shown in Fig. D.3.

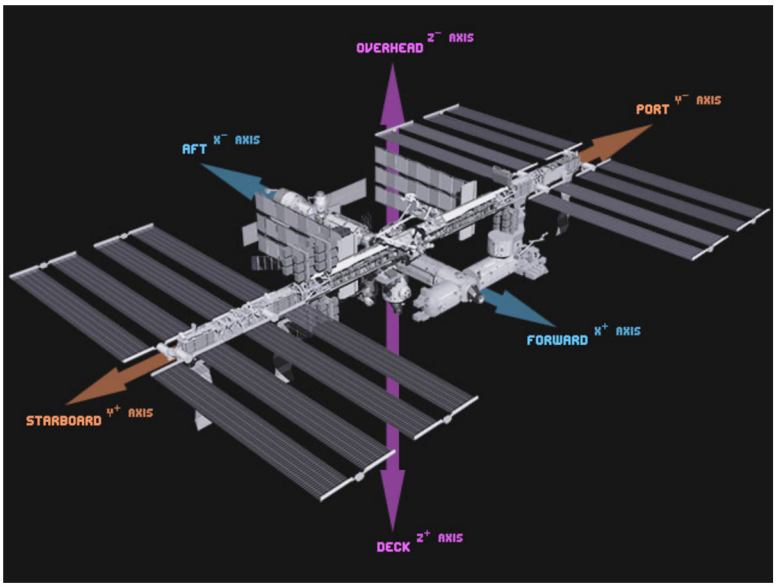


Figure D.3 International Space Station frame. Image Courtesy of NASA.

D.5. Selenographic frame

The Selenographic frame is shown in Fig. D.4. The Moon's prime meridian is the line passing from the lunar North pole to the South pole through the point on the lunar surface directly facing Earth. ϕ'_m is the Selenographic Latitude, which is the acute angle measured normal to the Moon's equator between the equator and a line connecting the geometrical center of the coordinate system with a point on the surface of the Moon. The angle range is between -90 and +90 degrees. λ_m is the Selenographic Longitude, which is the angle measured towards the West in the Moon's equatorial plane, from the lunar prime meridian to the object's meridian. The angle range is between 0 and 360 degrees. North is the section above the lunar equator containing Mare Serenitatis. West is measured towards Mare Crisium.

The relationship to the ECI frame is shown in Fig. D.5. i is the inclination between the Moon's equator and the ECI equator. Ω' is the longitude of the mean ascending

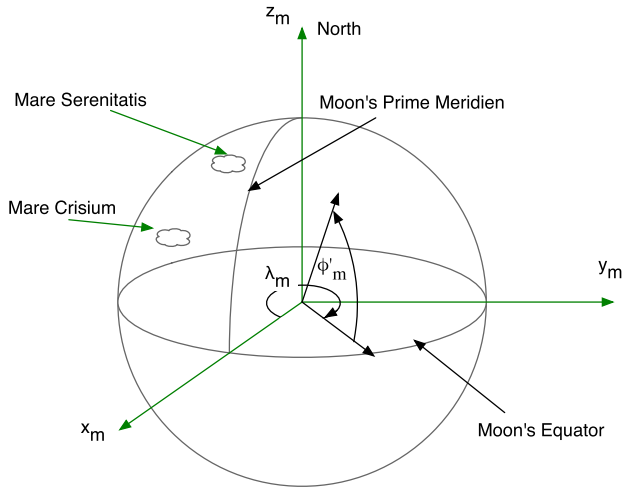


Figure D.4 The selenographic reference frame.

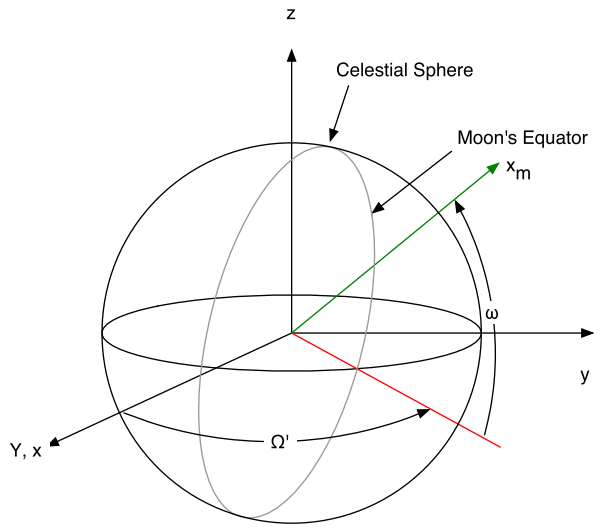


Figure D.5 Selenographic to ECI frame.

node of the lunar orbit referenced to the mean equinox. ω is the angle between the ascending node and the lunar prime meridian.

D.6. Areocentric (Mars) coordinates

The areocentric frame or Mars frame is shown in Fig. D.6.

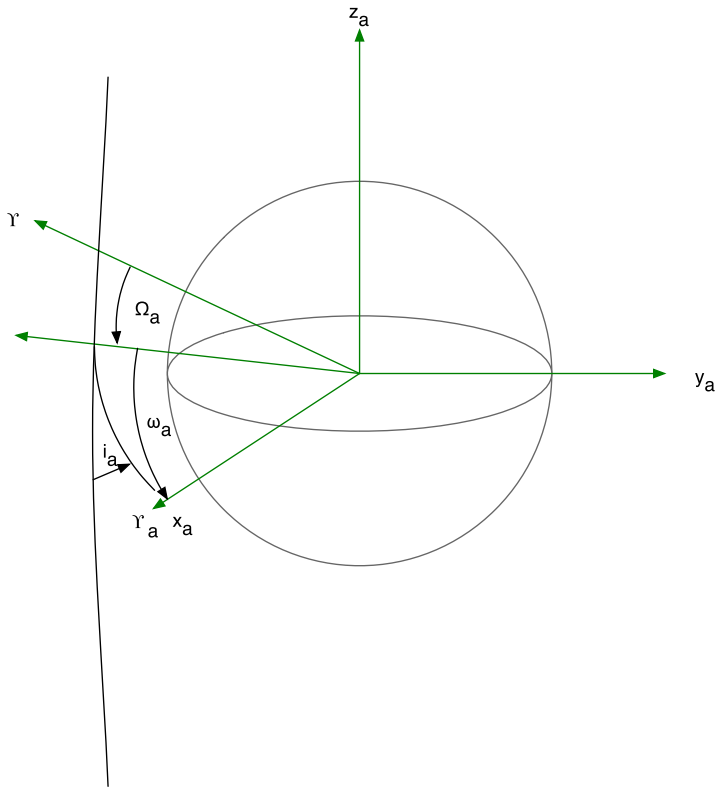


Figure D.6 Areocentric frame.

Ω_a, ω_a, and i_a are the standard Euler rotations of the Mars vernal equinox, Υ_a with respect to the Earth's vernal equinox, Υ.

APPENDIX E

Ephemeris

E.1. Introduction

The ephemerides give the positions and orientations of the planets and minor planets in the solar system. While typically used mostly by the mission-planning groups, they are also needed for attitude control. If a spacecraft must point at a target on a planet or moon, it needs to know the orientation of that body with respect to the inertial frame.

E.2. Planetary orbits

Planetary and moon orbits are necessary for stages of mission when the spacecraft approaches a target. Low-accuracy ephemerides are useful for preliminary work.

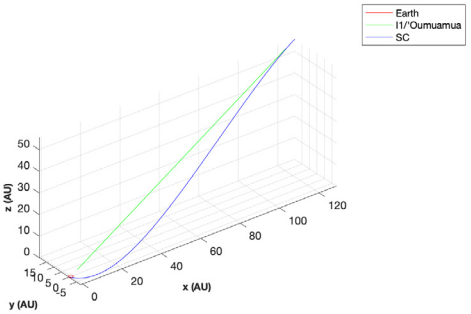
High-accuracy tools are available online or can be downloaded from NASA and other sources. A particularly useful high-accuracy tool is the JPL SPICE library [1]. This provides functions to compute the positions and velocities of the planets (including the nonplanet Pluto), the Earth's Moon, and the Sun.

An alternative is to use state vectors from JPL Horizons. JPL Horizon is an online solar-system data and ephemeris computation providing access to key solar-system data. It permits the production of highly accurate ephemerides for solar-system objects including (at the time of this book's publication) 1 124 802 asteroids, 3751 comets, 209 planetary satellites, 8 planets, the Sun, the first and second Lagrange points (L1 and L2), select spacecraft, and system barycenters [2]. Barycenters are one of the focii of an elliptical orbit. Example E.1 shows a nuclear-fusion-propelled spacecraft playing catch-up with I1/'Oumuamua, a recent interstellar visitor. The Horizons state vector that initializes the I1/'Oumuamua state is

```
jD = 2462576.5; % A.D. 2030-Mar-16 TDB
rA  = [ 1.008272975334536E+10;  1.579428200546252E+09;
    4.311503526924462E+09];
vA   = [ 2.428440175149515E+01;  3.623455975379303E+00;
    1.063812980138126E+01];
```

The units are km and km/s. The fusion-rocket interception trajectory is found by trajectory optimization. Trajectory propagation uses a simple point-mass Sun gravitational model. The trajectory might be used if one were to design a spacecraft to chase an interstellar object.

```
%% 11/'Oumuamua from
jD          = 2462576.5;% A.D. 2030-Mar-16 00:00:00.0000 TDB
rA          = [ 1.008272975334536E+10; 1.579428200546252E+09; 4.311503526924462E+09];
vA          = [ 2.428440175149515E+01; 3.623455975379303E+00; 1.063812980138126E+01];
mu          = 1.3271e+11;
el0         = RV2El(rA,vA,mu);
[r,v]       = RVOrbGen(el0,linspace(0,-12*365*86400,200),[],mu);
au          = 14959787;
Plot3D(r/au)

% Assume we could launch closer to when asteroid is detected (2018)
el          = RV2El(r(:,end),v(:,end),mu);
jD0         = jD - 12*365;
```

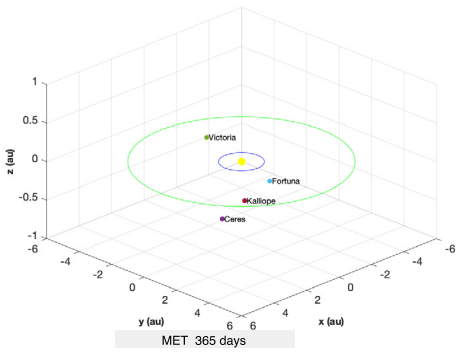


Example E.1: 11/'Oumuamua and a nuclear-fusion-powered spacecraft playing catch-up with the interstellar object.

E.3. Asteroid orbits

Asteroid orbits can also be found from the JPL Horizons System. Another option is The Asteroid Orbital Elements Database [3]. An example from this database is shown in Example E.2. Orbit propagation in the plot uses Keplerian propagation.

```
asteroids = {'ceres' 'victoria' 'Fortuna' 'Kalliope'};
ReadAsteroidDatabase( asteroids , 'astorbshort.dat' );
```



Example E.2: Four asteroids showing their heliocentric orbits. The orbit propagation is Keplerian.

E.4. Planetary orientation

Planetary orientation is determined by three angles:

1. Rotation angle about the planet's axis of rotation;
2. Nutation of the axis of rotation;
3. Precession of the axis of rotation.

The Earth-fixed and Earth-centered inertial (ECI) coordinate frames are illustrated in Fig. E.1.

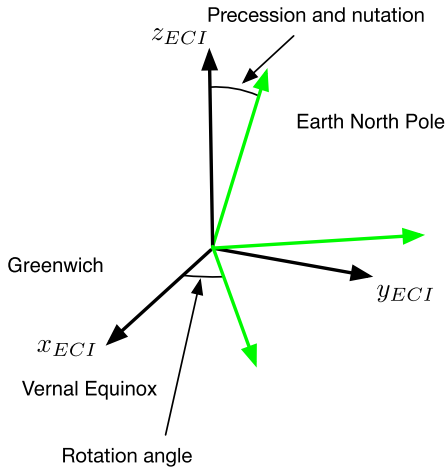


Figure E.1 ECI and Earth-fixed coordinate frames. In many applications, only the rotation angle needs be considered. The green arrows are the Earth-fixed frame.

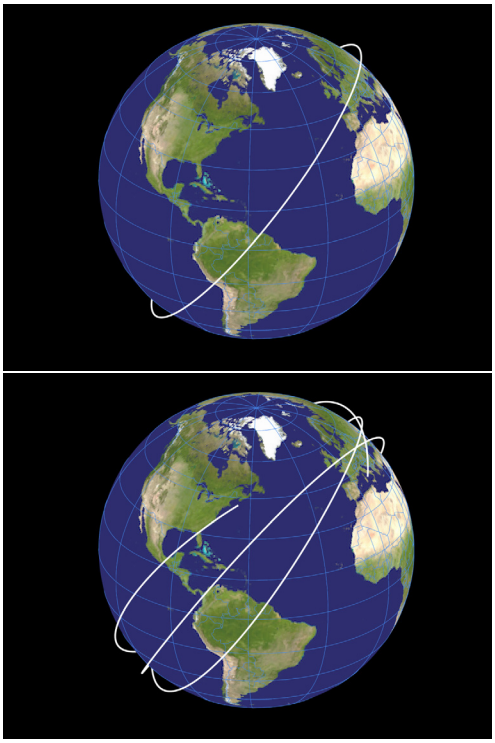
The rotation angle is the daily rotation of the planet. This is the dawn-to-dusk cycle we experience on the Earth. Precession is the slow, gravity-induced rotation of the axis. The Earth's precession is a 26 000-year cycle. Nutation is the short-period oscillation of the plane's axis. The three angles determine the orientation of the planet with respect to the inertial frame. If a planet-pointing spacecraft is using an inertial reference, such as a star camera, it needs the transformation matrix generated by these three angles to know how to relate a location on the planet with the orientation measured with the star camera.

When the Earth's magnetic field is used for attitude determination, it is also necessary to transform the magnetic field into the inertial frame. Example E.3 shows the International Space Station orbit in the ECI and the Earth-fixed frames in three dimensions. The rotation of the Earth is evident from the orbit track.

E.5. Asteroid dynamics

An asteroid's orientation may not fit into the rotation, precession, and nutation angle model of the major planets and moons. An asteroid may be rotating about all three axes in an irregular fashion. The angular rates may be coupled due to the inertia matrix of the asteroid not being that of a spherical body. For example, Example E.4 shows the rotation of the asteroid 4179 TOUTATIS [4]. The inertia matrix is normalized by I_{zz}. This does not change the dynamics since both terms are functions if I. The inertia of

```
%% Compute the ISSOrbit

[el, jD0] = ISSOrbit;
p         = Period(el(1));
t         = linspace(0,3*p,1000);
r         = RVOrbGen(el,t);

PlotOrbit(r,t,jD0)
PlotOrbit(r)
```



Example E.3: ISS orbit about the Earth in the ECI and Earth-fixed frames.

TOUTATIS cannot be accurately calculated using the shape and mass. It is believed that the mass distribution is irregular.

$$I\dot{\omega} - \omega^{\times} I\omega = 0 \tag{E.1}$$

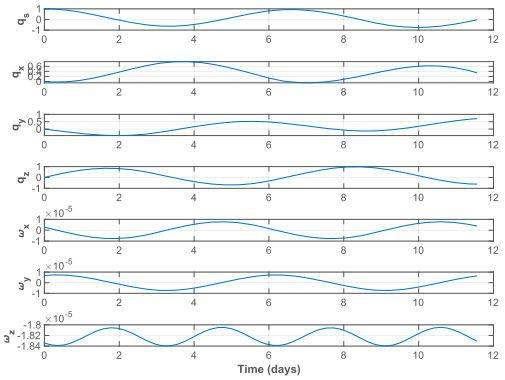
$$\frac{I\dot{\omega} - \omega^{\times} I\omega}{I_{zz}} = 0 \tag{E.2}$$

The asteroid is subject to torques due to the Earth, Sun, and Moon's gravity. The disturbance torques are not included in the simulation. To study the effect of solar torques, the actual inertias, not ratios, are needed.

E.6. Stars

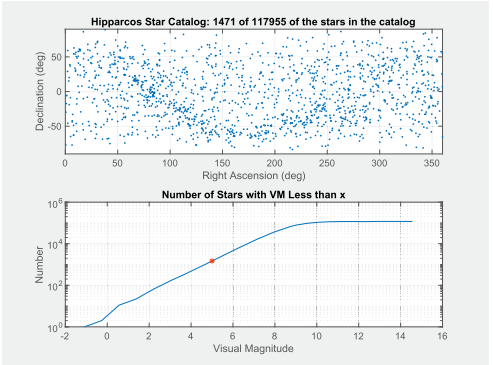
Star catalogs are needed for attitude determination using star cameras. There are many star catalogs. One well-known catalog is the Hipparcos catalog [5]. The European Space Agency's Hipparcos space astrometry mission catalogs over 100 000 stars. Example E.5 displays the catalog.

```
1
2  omega = [14.14;33.532;-90.793]*pi/180/86400;
3  x     = [1;0;0;0;omega];
4  n     = 1000;
5  xP    = zeros(7,n);
6
7  dT    = 1000;
8  for k = 1:n
9    xP(:,k) = x;
10   x = RK4(@RHS,x,dT,0);
11 end
12
13 yL = {'q_s' 'q_x' 'q_y' 'q_z' '\omega_x' '\
       omega_y' '\omega_z'};
14 TimeHistory((0:n-1)*dT,xP,yL,'TOUTATIS');
15
16 function xDot = RHS(x,~)
17
18 inr = [ 3.0836  -7.1e-4  1.17e-3;...
19        -7.1e-4   3.235   1.3e-3;...
20         1.2e-3   1.3e-3  1];
21
22
23 omega = x(5:7);
24 q     = x(1:4);
25
26 qDot  = QIToBDot(q,omega);
27 omegaDot = -inr\cross(omega,inr*omega);
28
29 xDot = [qDot;omegaDot];
30
31 end
```



Example E.4: 4179 TOUTATIS attitude motion. The motion is irregular. The inertia matrix is scaled by I_{zz}.

```
1 LoadCatalog
```



Example E.5: The Hipparcos star catalog. The bottom plot shows the number of stars as a function of visual magnitude. Visual magnitude is the brightness perceived by the human eye. Beyond ten the number of stars increases slowly.

When using the catalog for high-accuracy attitude determination, corrections must be made for:

1. radial velocity of the star;
2. parallax;
3. gravitational light deflection;

4. relativistic aberration due to planet and spacecraft velocity.

This process is known as stellar reduction [6].

A newer source for star data is the Gaia archive [7]. Gaia is a European space mission providing astrometry, photometry, and spectroscopy of more than a billion stars. It also has data for extragalactic and solar-system objects.

References

[1] C. Acton, N. Bachman, B. Semenov, E. Wright, A look towards the future in the handling of space science mission geometry, in: Enabling Open and Interoperable Access to Planetary Science and Heliophysics Databases and Tools, Planetary and Space Science 150 (2018) 9–12.
[2] R.S. Park, HORIZONS System, Sep 2021.
[3] T. Bowell, The asteroid orbital elements database, http://naic.edu/~nolan/astorb.html, Feb 2007.
[4] Y. Takahashi, M.W. Busch, D.J. Scheeres, Spin state and moment of inertia characterization of 4179 Toutatis, The Astronomical Journal 146 (4) (sep 2013) 95.
[5] M.A.C. Perryman, L. Lindegren, J. Kovalevsky, E. Hog, U. Bastian, P.L. Bernacca, M. Creze, F. Donati, M. Grenon, M. Grewing, F. van Leeuwen, H. van der Marel, F. Mignard, C.A. Murray, R.S. Le Poole, H. Schrijver, C. Turon, F. Arenou, M. Froeschle, C.S. Petersen, The Hipparcos Catalogue, Astronomy & Astrophysics 500 (1997) 501–504.
[6] P.K. Seidelmann, Explanatory Supplement to the Astronomical Almanac, University Science Books, Mill Valley, CA, 1992.
[7] Welcome to the Gaia Archive at ESA, https://gea.esac.esa.int/archive/.

APPENDIX F

Laplace transforms

F.1. Using Laplace transforms

Laplace transforms are a convenient way to solve time-invariant linear equations. These often arise in control-system design. In many cases the full nonlinear system can be linearized about an operating point and the resulting system is often time invariant. Laplace transforms can be used for such systems.

The definition of the one-sided Laplace transform is

$$F(s) = \{f(t)\} = \int_{0-}^{\infty} f(t)e^{-st}dt \tag{F.1}$$

The lower integral limit, 0- means that it includes 0 so that impulses may be modeled. An impulse is an input of infinitesimal duration. For solving differential equations the most important properties are differentiation.

$$\dot{f} = sF(s) - f(0-) \tag{F.2}$$

and second differentiation

$$\ddot{f} = s^2F(s) - sf(0-) - \dot{f}(0-) \tag{F.3}$$

For example, assume that we have the second-order differential equation

$$\ddot{x} + x = u \tag{F.4}$$

$$x(0) = x_0 \tag{F.5}$$

$$\dot{x}(0) = \dot{x}_0 \tag{F.6}$$

The Laplace transform of the differential equation is

$$s^2X(s) - sx(0-) - x(\dot{0}-) + X(s) = u(s) \tag{F.7}$$

or

$$X(s) = \frac{u(s)}{s^2+1} + \frac{sx_0 + \dot{x}_0}{s^2+1} \tag{F.8}$$

There are two parts to the response. One is due to the initial conditions, $x(0)$ and $\dot{x}(0)$, and the other due to the input.

Two special inputs are the step and the impulse. In the case of a step

$$u(s) = \frac{u_0}{s} \tag{F.9}$$

With a step, the input is at a constant value from time 0 to time infinity. For an impulse

$$u(s) = u_0 \tag{F.10}$$

The input is nonzero only at time 0. An impulse (sometimes called the Dirac-delta function, δ), is an input with an amplitude of u_0 but that happens in an infinitesimally short time. In this case the step response with zero initial conditions is

$$X(s) = \frac{u_0}{s(s^2+1)} \tag{F.11}$$

and the impulse response is

$$X(s) = \frac{u_0}{s^2+1} \tag{F.12}$$

Using a table of Laplace transforms the solution for the impulse response in the time domain is

$$x(t) = u_0 \sin t \tag{F.13}$$

For the step we first expand into partial fractions

$$X(s) = \frac{as+b}{s^2+1} + \frac{c}{s} \tag{F.14}$$

The numerator is always the order s one less than the denominator.

We need to match terms so we put this over a common fraction

$$X(s) = \frac{cs^2 + c + as^2 + bs}{s(s^2+1)} \tag{F.15}$$

Matching terms

$$c = u_0 \tag{F.16}$$

$$a = -u_0 \tag{F.17}$$

$$b = 0 \tag{F.18}$$

or

$$X(s) = \frac{u_0}{s} - \frac{su_0}{s^2+1} \tag{F.19}$$

Again using a table of transforms, the solution for the step input is

$$x(t) = u_0(1 - \cos t) \tag{F.20}$$

The first term comes from the $\frac{u_0}{s}$ and the second from the $\frac{su_0}{s^2+1}$ term. We could have solved for the response to the initial conditions the same way. Usually, we use the Laplace transform to solve for the response to inputs, but as shown above it can always be used to solve the response to both initial conditions and inputs.

F.2. Useful transforms

Table F.1 gives a short list of Laplace transform pairs useful for control work.

Table F.1 Laplace transform pairs.

Transform Pair	$F(s)$	$f(t)$
Single Differentiation	$sF(s) - f(0+\epsilon)$	$\frac{df(t)}{dt}$
Multiple Differentiation	$s^nF(s) - \sum_{k=1}^{n} s^{n-k}\frac{d^{k-1}f}{dt^{k-1}}(0+\epsilon)$	$\frac{d^nf(t)}{dt^n}$
Integration	$\frac{F(s)}{s} + \frac{\int f(t)dt}{s}$	$\int f(t)dt$
Linearity	$aF(s)$	$af(t)$
Time translation	$e^{\tau s}F(s)$	$f(t+\tau)$
S translation	$F(s+\frac{1}{T})$	$e^{\frac{-t}{T}}f(t)$
Impulse	1	$\delta(t)$
Doublet	s	$\dot{\delta}(t)$
Step	$\frac{1}{s}$	1
Lag	$\frac{T}{Ts+1}$	$e^{-\frac{t}{T}}$
Sine	$\frac{\omega_n}{s^2+\omega_n^2}$	$\sin\omega_n t$
Cosine	$\frac{s}{s^2+\omega_n^2}$	$\cos\omega_n t$
Second-order sine	$\frac{1}{s^2+2\zeta\omega+\omega_n^2}$	$\frac{1}{\omega_n\sqrt{1-\zeta^2}}e^{-\zeta\omega_n t}\sin\omega_n\sqrt{1-\zeta^2}t$
Second-order cosine	$\frac{s}{s^2+2\zeta\omega+\omega_n^2}$	$\frac{1}{\sqrt{1-\zeta^2}}e^{-\zeta\omega_n t}\cos(\omega_n\sqrt{1-\zeta^2}t+\theta)$

APPENDIX G

Control theory

G.1. Introduction

Control theory relies on the concept of feedback. Feedback is used to:

1. Stabilize a system or improve the stability of a system. For example, a system that oscillates is considered stable (in the sense that the system remains bounded). A feedback control system can increase the damping in the system so that the oscillations persist for only a short time.
2. Reduce the sensitivity of a system to modeling errors. Open-loop control could be used if the system were perfectly known. Feedback corrects for errors in tracking, thus making the closed-loop system behave closer to the ideal.
3. Reject disturbances and attenuate noise. This is done by limiting the response of the system at frequencies where there is noise (usually high frequencies) and by tightly controlling the system (applying significant restoring control) where there are disturbances.
4. Change the transient response. Feedback can speed up or slow down the response as needed. Oscillatory behavior can be reduced.

This chapter describes control-system design. The fundamental approach used throughout is loop shaping. This is applied both for continuous and discrete-time control systems. We motivate the problem in the next section by applying control-system concepts to a very simple system.

G.2. Simple control system

Assume that you want to control the angular rate of a rotor. Further, assume that you can measure that rate with a perfect rate sensor and have a perfect torque actuator. Neither can saturate so the sensor can measure arbitrarily high rates and the torque actuator can produce arbitrarily large torques. In addition, neither sensor nor actuator has any noise and both have infinite resolution. The dynamical system is

$$\begin{aligned} I\dot{\omega} &= T_c + T_d \\ y &= \omega \\ u &= \omega_c \end{aligned} \tag{G.1}$$

where y is the measured value of rate, u is the command, T_d is the disturbance torque, and T_c is the control torque. A simple control law is

$$T_c = -k(\omega - \omega_c) \tag{G.2}$$

where the forward gain k is in units of N m/rad/s. Combining the equations, we get the closed-loop dynamical equation

$$I\dot{\omega} + k\omega = k\omega_c + T_d \tag{G.3}$$

We can solve this analytically to see how well the control system performs. In this case we are interested in disturbance rejection and command following. First, divide by the inertia

$$\dot{\omega} + \frac{k}{I}\omega = \frac{k}{I}\omega_c + \frac{T_d}{I} \tag{G.4}$$

The unforced (homogeneous) solution is

$$\omega_h = Ae^{-\frac{k}{I}t} \tag{G.5}$$

The forced (particular) solution is

$$\omega_p = \omega_c + \frac{T_d}{k} \tag{G.6}$$

Given the initial condition ω_0 the total solution to a simultaneous step in command and step torque disturbance is

$$\omega = (\omega_0 - \omega_c - \frac{T_d}{k})e^{-\frac{k}{I}t} + \omega_c + \frac{T_d}{k} \tag{G.7}$$

The rate will approach the commanded rate based on the damping time constant that is proportional to k. The disturbance is attenuated by the factor k. As a consequence, a large k is better. However, large k implies large torques for small errors, meaning our actuator has to produce a lot of torque and will consume a lot of energy.

This simple system can be represented as a block diagram (see Fig. G.1). The block diagram makes use of Laplace-transform notation, where $1/s$ implies integration. If we write the open-loop equations in Laplace-transform notation we get the equations

$$\omega(s) = \frac{1}{Is}(T_d(s) + k(\omega_c(s) - \omega(s))) \tag{G.8}$$

$$\omega(s)\left(1 + \frac{k}{Is}\right) = \frac{T_d(s)}{Is} + \frac{k\omega_c(s)}{Is} \tag{G.9}$$

$$\omega(s) = \frac{T_d(s)}{Is + k} + \frac{k\omega_c(s)}{Is + k} \tag{G.10}$$

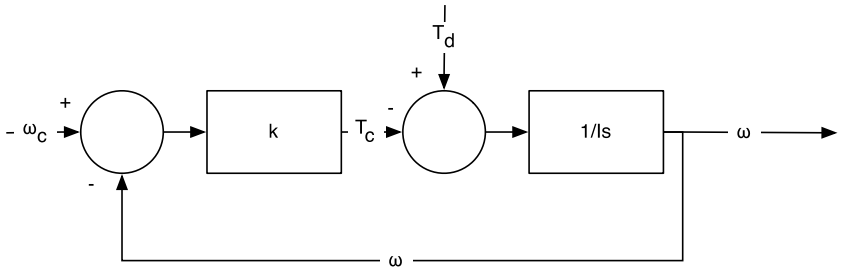
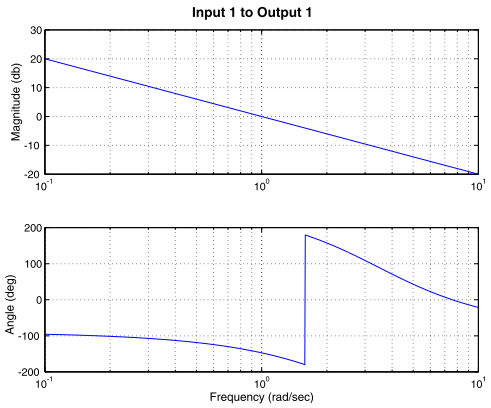


Figure G.1 Rate-control block diagram.

This example can be studied using Bode plots, Nichols diagrams, and root-locus plots. The state-space equations for the plant and the controller when we are looking at the disturbance input are

$$\begin{array}{llll} a=0 & b=1/I & c=1 & d=0 \\ a_c=0 & b_c=0 & c_c=0 & d_c=k \end{array} \tag{G.11}$$

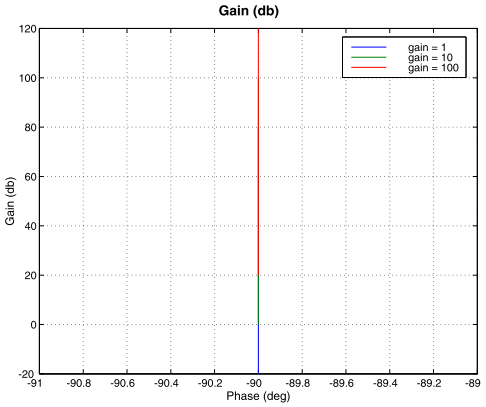
```
gain = [1 10 100];
w    = logspace(-2,3);
for k = 1:3
  [a,b,c,d] = Series(0, 1, 1, 0, [], [], [], gain
      (k) );
  [mag(k,:),ph(k,:)] = FResp(a,b,c,d,1,1,w);
end
yL = {'Gain (db)', 'Phase (deg)'};
h = Plot2D( w, [20*log10(mag);ph], 'Frequency (
      rad/s)',...
           yL, 'Bode Plot', 'xlog', {'[1:3]' '
                [4:6]'})
legend('gain = 1', 'gain = 10', 'gain = 100')
set(h,'color',[1 1 1]);
```



Example G.1: Bode plot.

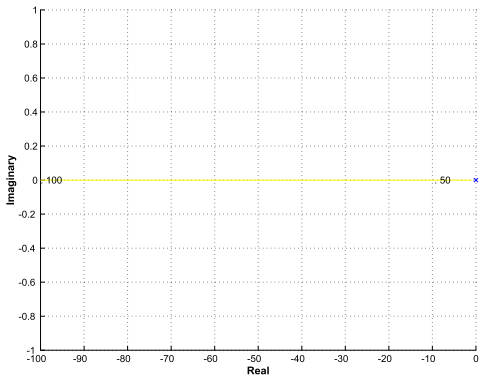
Example G.1 shows the open-loop equations when $I = 1$ and $k = 1, 10, 100$. Bode and Nichols plots show the same information. The open-loop phase shift is constant at -90 deg. As the gain increases, the closed-loop gain increases. This moves the Bode plot up in the plot, thus increasing the crossover frequency, which is the frequency beyond which feedback has no influence on the closed-loop response. As the phase shift is always -90° the system is stable for all gains. In real systems this is never true because all real systems have delays and any delay will cause the system to go unstable for some value of the gain. Example G.3 shows the root-locus plot for k = 0 to 100 for the same system. The root-locus plot gives the closed-loop response. This plot shows that as the gain

```
gain = [1 10 100];
for k = 1:3
  [a,b,c,d] = Series(0, 1, 1, 0, [], [], [], gain
      (k) );
  [mag(k,:),ph(k,:)] = FResp(a,b,c,d);
end
h = Plot2D( ph, 20*log10(mag), 'Phase (deg)', '
    Gain (db)');
legend('gain = 1', 'gain = 10', 'gain = 100')
set(h,'color',[1 1 1]);
```



Example G.2: Nichols plot.

```
[a,b,c,d] = Series(0, 1, 1, 0, [], [], [], 1 );
g         = statespace(a,b,c,d);
gain      = logspace(0,2);
RootLocus( g, gain );
PrintFig(1,1,1,'RootLocus')
```



Example G.3: Root-locus plot.

goes up the closed-loop pole becomes more negative real, i.e., it becomes faster. The imaginary part remains zero. More information on single-input–single-output design is given later in this chapter.

G.3. The general control system

Fig. G.2 shows a general block diagram of a control system. r is the command input to the system, m is the measurement noise input, y is the system-measured output, d_1 is the disturbance that is measured and used for feedforward purposes, d_2 is the unmeasured disturbance, R is the command prefiltering block, K is the control block, Q is the feedback block, F is the feedforward block, and P is the plant. P includes all plant dynamics including those of the actuators and sensors. If the system is implemented

digitally, K and F include samplers on the inputs and sample-and-holds, or pulsewidth modulators, etc. on the outputs. E is the identity matrix.

Often, unity feedback configurations are chosen in which $Q = 1$. If r is always 0 then the system is a regulator and R is zero. Many control systems do not have disturbance feedback and $F = 0$. Sometimes the disturbances inputs are put after the plant and the relevant plant dynamics are included in the disturbances.

This block diagram applies equally well to systems with multiple inputs and outputs. It is important to remember that when dealing with the multioutput system (which all real systems are), not all outputs are measured by control sensors, but may still be important to the performance.

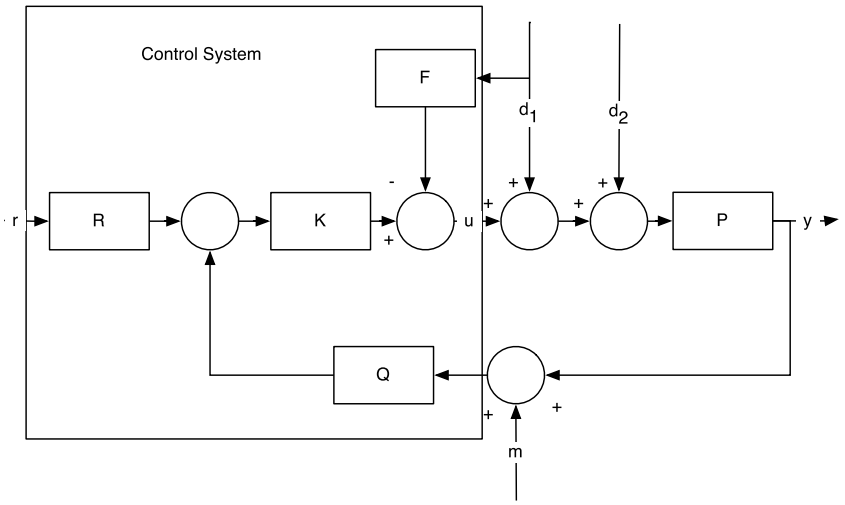


Figure G.2 General control-system block diagram.

G.4. Fundamental relationships

In this section we will derive algebraic relationships between the inputs and outputs of Fig. G.2. The error e is

$$e = Rr - Q(m + y)) \tag{G.12}$$

the output, y is

$$y = P((EW - F)d + u) \tag{G.13}$$

and the control is

$$u = Ke \tag{G.14}$$

The system inputs are m, r, and d. u and e are intermediate quantities. Solving for y in terms of the inputs

$$(E + PKQ)y = P(E - F)d + PK(Rr - Qm) \tag{G.15}$$

The feedforward matrix attenuates the disturbance d and may modify its dynamics. For example, if d were a step input and F were a low-pass filter, d would then look like the product of a step and an exponential decay. Since F cannot equal E without drawing infinite power, some of d will always pass through. Therefore no generality is lost if F is dropped from further discussion since the magnitude and dynamics of d are not significant in this discussion.

Q modifies the characteristics of the feedback path. If we set

$$K = KQ^{-1} \tag{G.16}$$

and

$$R = QR \tag{G.17}$$

then Q disappears. Q may give some design flexibility when mechanizing the control system, but from a strictly algebraic point-of-view can be eliminated.

Eliminating Q and F and multiplying both sides by $E + PK$ gives

$$y = -(E + PK)^{-1}(Pd + PK(Rr - m)) \tag{G.18}$$

where

$$F_o = E + PK \tag{G.19}$$

is known as the output-return difference. This is because if the feedback loop were broken at the output of the plant, P, then F_o would be E, the identity matrix. The return difference shows the effect of feedback on the output of the plant. In Eq. (G.18) note that r and m are indistinguishable to the output, except that r is multiplied by the prefiltering matrix R.

The inverse of (G.19) is known as the sensitivity, S. It shows how sensitive the output is to the product of the plant dynamics and the disturbances. If the sensitivity is small, the output will be insensitive to disturbances. If it is one, the product of the plant dynamics and the disturbances will be passed unattenuated to the output. The sensitivity is one when the loop is opened so the sensitivity (at any given frequency) shows how close the system is to being open loop at that frequency. Eq. (G.18) can be rewritten as

$$y = Spd + G_c r - Tm \tag{G.20}$$

where G_c is the transmission function and equals

$$G_c = TR \tag{G.21}$$

and determines how well y will track r. Ideally, G_c would equal E, by letting

$$R = T^{-1} \tag{G.22}$$

but again that would require infinite power and is not possible. T is the closed-loop transfer function

$$T = SPK \tag{G.23}$$

and determines how well the noise is attenuated. Ideally it should equal zero. It is sometimes called the complementary sensitivity because

$$S + T = E \tag{G.24}$$

where

$$S = (I + PK)^{-1} \tag{G.25}$$

and

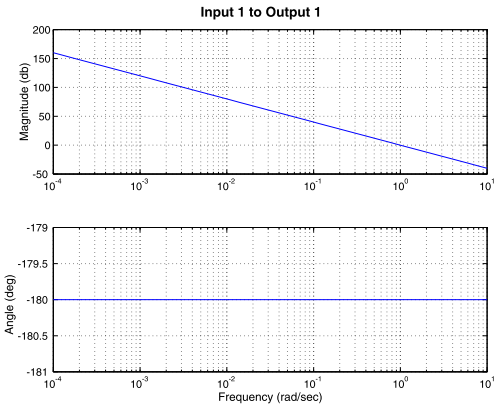
$$T = PK(I + PK)^{-1} \tag{G.26}$$

This leads to the fundamental trade-off in control design. Ideally, S and T would be equal to zero everywhere but since their sum is the identity matrix, this is not possible. At best, and what control design is all about, S can be made small where P_d is large and T small where m is large. This works in practice since P_d is usually large at low frequencies and small at high frequencies, while sensor noise is generally high at high frequencies, and low at low frequencies except for a component near dc (known as a bias) which varies slowly with time. Table G.1 summarizes the relationship between PK, T, and S. Example G.4 shows a double-integrator plant, which represents a single axis of a vehicle in its simplest form. At $\omega = 0$ it has infinite gain and the gain rolls off as $1/\omega^2$. It is clear how important P is when evaluating the effect of disturbances. In this case, any steady-state disturbance will eventually cause the output to grow very large.

Table G.1 Relationships between *PK*, *T*, and *S*.

	PK **small**	*PK* **large**
T	0	E
S	E	0

```
a = [0 1;0 0]; b = [0;1];
c = [1 0]; d = 0;
FResp(a,b,c,d,1,1,logspace(-4,1))
```



Example G.4: Double-integrator plant.

It is also important to keep u small, since that determines how much power is consumed, how large the actuators must be, etc. The control output is

$$u = (E + KP)^{-1}K(Rr - m - Pd) \tag{G.27}$$

The factor $E + KP$ is called the input return difference (denoted by F_i) since it shows the effect of feedback on the input to the plant from the control. Note that K and P are transposed from the output return difference function. If the plant is single-input–single-output K and P are scalars so the order does not matter. For multiinput–multioutput plants, it does make a difference. If $KP \gg E$ Eq. (G.27) becomes

$$u = P^{-1}(Rr - m) - d \tag{G.28}$$

This says that if the control is fast enough, it will exactly cancel the disturbances, d. The reference and the noise inputs will be amplified by the inverse of the plant. Thus in the double-integrator example, high-frequency references and noise will require more command energy than low-frequency ones.

Finally, look at the tracking error, which is defined as

$$e = r - y \tag{G.29}$$

Substituting,

$$e = (E - TR)r - (E - T)Pd + Tm \tag{G.30}$$

where T should equal E to eliminate tracking errors due to disturbances, but that would cause all noise to pass through. TR should equal E to eliminate tracking errors due to the reference input. Of course, for TR to equal E, R would have to be the inverse of T,

the closed-loop transfer function, which is not physically possible. Nonetheless, these goals can be achieved over different parts of the frequency spectrum. Thus at frequencies where the reference is likely to have energy TR should equal E. At disturbance frequencies, T should equal E, and everywhere else T should equal zero to minimize noise, see Table G.2.

Table G.2 What quantities should be small to compensate for large inputs. The quantities in the table make the quantities on the left small.

	m large	P_d large	r large
y	T	S	N/A
e	T	$E-T$	$E-TR$
u	F_iK	F_iK	F_iKR

G.5. Tracking errors

In Eq. (G.30) if $R = E$ then the transfer function between the reference input and the error is just the sensitivity S. Rr is just a modification of the input-signal dynamics, therefore the sensitivity will determine the tracking error to the output of r.

Disturbances and reference signals can be characterized in the frequency domain just as plants and controllers can. Typical inputs are given with their transfer functions in Table G.3. R can be used to modify the dynamics of the inputs. This can make it

Table G.3 Inputs.

Input	Time Domain	Frequency Domain
Impulse	d	1
Step	A	A/s
Ramp	At	A/s^2
Sinusoid	$A\sin\omega t$	$A\omega/(s^2+\omega^2)$

easier for the closed-loop system to follow its input.

For example, given the simple single-input–single-output plant with a gain of k, a $1/s$ plant, an input transfer function of R and a ramp input, the error is

$$\frac{e}{A} = \frac{R}{s^2}\left(\frac{s}{s+k}\right) \tag{G.31}$$

where the quantity in parentheses is the sensitivity. The steady-state tracking error can be found using the final-value theorem, which states that

$$\lim_{t\to\infty} f(t) = \lim_{s\to 0} sf(s) \tag{G.32}$$

R must be of order s for the tracking error to go to zero in steady state. A suitable transfer function would be

$$R = \frac{s}{\tau s + 1} \tag{G.33}$$

An alternative is to make the controller a proportional-integral controller.

$$K = k\left(1 + \frac{1}{\tau s}\right) \tag{G.34}$$

Then, the sensitivity would become

$$S = \frac{\tau s^2}{\tau s^2 + k\tau s + 1} \tag{G.35}$$

which has the same desired effect. In general, for the tracking error to approach zero asymptotically, the sensitivity must be of the same order as the input.

G.6. State-space closed-loop equations

Fig. G.3 shows the block diagram in the unity feedback configuration without the command prefiltering.

The control block and the plant can both be represented by sets of first-order differential equations

$$\dot{x}_p = A_p x_p + B_p u_p \tag{G.36}$$
$$y_p = C_p x_p + D_p u_p \tag{G.37}$$
$$\dot{x}_c = A_c x_c + B_c u_c \tag{G.38}$$
$$y_c = C_c x_c + D_c u_c \tag{G.39}$$
$$u_c = r - m - y_p \tag{G.40}$$
$$u_p = d + K y_c \tag{G.41}$$

The state equations for the closed-loop plant are

$$\dot{x} = Ax + B_d d + B_m(r - m) \tag{G.42}$$

and the measurement equation is

$$y = Cx + D_d d + D_m(r - m) \tag{G.43}$$

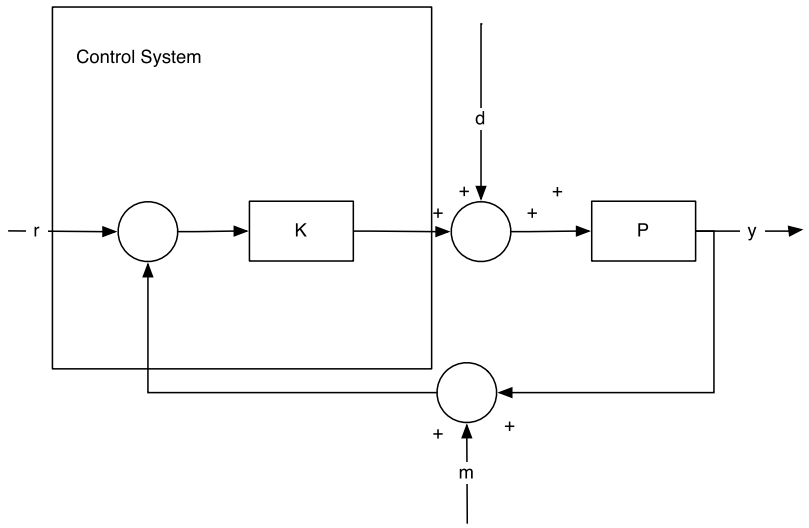


Figure G.3 Control-system block diagram.

where

$$x = \begin{bmatrix} x_p \\ x_c \end{bmatrix} \tag{G.44}$$

$$A = \begin{bmatrix} A_p - B_p K D_c F C_p & B_p K (C_c - D_c F D_p K D_c) \\ -B_c F C_p & A_c - B_c F D_p K C_c \end{bmatrix} \tag{G.45}$$

$$B_d = \begin{bmatrix} B_p (E - K D_c F D_p) \\ -B_c F D_p \end{bmatrix} \tag{G.46}$$

$$B_m = \begin{bmatrix} B_p K D_C (F D_p K D_c - E) \\ -B_c (E - B_p K D_c F D_p) \end{bmatrix} \tag{G.47}$$

$$C = \begin{bmatrix} F C_p \\ F D_p K C_c \end{bmatrix} \tag{G.48}$$

$$D_d = F D_p \tag{G.49}$$

$$D_m = -F D_p K D_c \tag{G.50}$$

$$F = (E + D_p K D_c)^{-1} \tag{G.51}$$

Adding the prefilter plant is straightforward. r is replaced by y_r and an additional set of state equations

$$\dot{x}_r = A_r x_r + B_r r \tag{G.52}$$

$$y_r = C_r x_r + D_r r \tag{G.53}$$

$B_m C_r$ is added to the state equations for x_p and x_c. $B_m D_r$ becomes a new input matrix with the third block row equal to B_r. γ is unchanged. The final state equation becomes

$$\dot{x} = \begin{bmatrix} A_{pp} & A_{pc} & B_{mp}A_r \\ A_{cp} & A_{cc} & B_{mc}A_r \\ 0 & 0 & A_r \end{bmatrix} \begin{bmatrix} x_p \\ x_c \\ x_r \end{bmatrix} + \begin{bmatrix} B_{pm} & B_{pd} & B_{mp}D_r \\ B_{cm} & B_{cd} & B_{mc}D_r \\ 0 & 0 & D_r \end{bmatrix} \begin{bmatrix} m \\ d \\ r \end{bmatrix} \tag{G.54}$$

G.7. Approaches to robust control

G.7.1 Introduction

This section outlines several approaches to robust control. Robustness is the property of a closed-loop system by which it remains stable and delivers acceptable performance in the face of unmodeled perturbations to the plant, disturbances, and noise. These perturbations may be as simple as unmodeled gain variations in an actuator, or as complex as variations in the structure of the plant model. The two forms of robustness are known as stability and performance robustness. When discussed in the context of multivariable control systems, an additional important property, sometimes known as the integrity of the system, is its stability in the face of loop failures.

Since the field of robust control is very active, it is not possible to describe all methodologies. Noticeably absent in this section are detailed discussions of adaptive control and variable-structure control. Adaptive control has been used with a fair degree of success in the process-control industry, but does not, at present, appear suitable for fully autonomous control systems. Variable-structure control has been proposed for control systems and has been successfully applied to induction motor controllers, among other applications, but since it is a nonlinear control approach it requires extensive simulations to validate, making it costly to implement and impractical given the schedule constraints on most flight software.

G.7.2 Modeling uncertainty

Robust control techniques require models of uncertain dynamics. The form of the model will directly determine the conservativeness of the resulting control system. The more conservative a design is, the poorer its performance will be with the nominal dynamics; this can lead to a need for better, and more expensive, actuators and sensors.

In classical control design, uncertainty is often represented by loop-gain uncertainty at the phase-crossover frequencies and phase uncertainty at the gain-crossover frequencies. These translate into the gain and phase margins. Complex systems, such as those with bending or nutation modes near or within the control-system bandwidth, have multiple crossovers. For example, a designer of a single-input–single-output (SISO) system might decide that the frequency of the first bending-mode frequency was so uncertain that the loop-crossover frequency had to be significantly below the nominal

first bending-mode frequency so that the bending mode was gain stabilized. Gain stabilization implies that the mode peak is below 0 dB and no amount of phase shift can cause it to destabilize the closed-loop system. Naturally, this leads to a lower performance since the bandwidth is restricted and lower than the frequency of the bending mode. If the modal frequency were better known, a notch filter might be employed or the mode might be phase stabilized. These solutions would lead to better performance.

A standard representation of uncertainty is illustrated in Fig. G.4. All uncertainty in

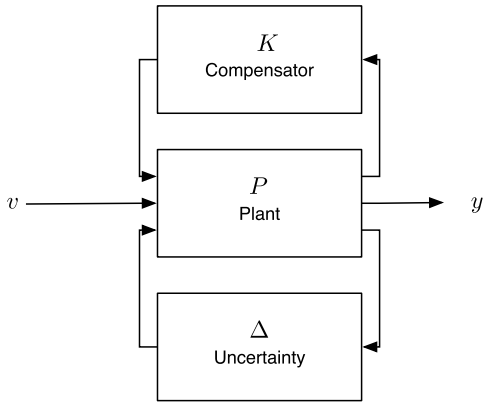


Figure G.4 Standard representation of uncertainty.

the plant is pulled out into the transfer matrix Δ. The simplest approach is to permit Δ to be unstructured and to only restrict its size that is measured by the matrix norm

$$||\Delta(s)||_\infty \tag{G.55}$$

This expression is the maximum value of the norm of $\Delta(s)$. For a single-input–single-output transfer function, it would be the peak magnitude on a Bode plot.

The size can also be made frequency dependent by pre- and postmultiplication by frequency-dependent weighting functions. The uncertainty may be treated as an additive perturbation to the plant or a multiplicative uncertainty. In the latter case, which is often more useful since the size of Δ can be specified as a percentage variation in the plant dynamics, the uncertainty must be specified as either an input or output multiplicative uncertainty since matrix multiplication does not commute.

When a robust design is based on the size of Δ, the system will remain stable regardless of the variations within Δ as long as the size remains within the prespecified limits. This can lead to a conservative design. For example, even though it may be known that Δ is diagonal, a given size of Δ can also accommodate off-diagonal terms. Therefore a design based on the size will tend to be conservative.

A less-conservative approach is to account for the structure of Δ. Structured uncertainty methods take advantage of knowing which elements of Δ are nonzero. Δ takes on a block structure with submatrices representing unstructured uncertainty. The uncertainty model might be

$$\Delta = \begin{bmatrix} \delta_r & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Delta_3 \end{bmatrix} \tag{G.56}$$

where Δ_3 represents the plant uncertainty and δ_r is the roll-channel gain uncertainty. The pitch-channel uncertainty is zero. This model, while better than the unstructured model, is still conservative because the elements of Δ may be known to be purely real, yet any general bound on Δ will permit complex elements. As a consequence, a design employing this form of structured uncertainty will accommodate phase uncertainty that may not be physically possible.

A more accurate representation of uncertainty is to compute the set of possible frequency responses of the plant at each frequency. For example, for a double integrator with a flexible mode and collocated sensor and actuator, the plant is

$$\frac{\Theta(s)}{T(s)} = \frac{s^2 + 2\zeta\omega s + \omega^2}{Is^2(s^2 + \gamma(2\zeta\omega s + \omega^2))} \tag{G.57}$$

where γ is the modal coupling factor

$$\gamma = 1 + \frac{I_\mu}{I} \tag{G.58}$$

where I_μ is the modal inertia and γ is ≥ 1. The uncertainty template would be found by computing all values of the transfer function for the specified range of the parameters. This permits a direct representation of the uncertainty with all of the gain and phase information. Generally, I and ω are known accurately, but γ and ζ are not. Nonetheless, the transfer function varies in a deterministic way with these parameters and it is easy to find a set of uncertain plants for realistic ranges of γ and ζ. There is nothing to prevent the use of a nonlinear relationship between the gain, phase, and plant parameters.

The second form of robustness, performance robustness, can be related to the structured uncertainty described above. Given the model in Fig. G.5 the transfer function is

$$\begin{bmatrix} \gamma \\ x \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} v \\ z \end{bmatrix} \tag{G.59}$$

and performance robustness can be defined as

$$||Q_{11} + Q_{12}\Delta(I - Q_{22}\Delta)^{-1}Q_{21}||_\infty < 1 \tag{G.60}$$

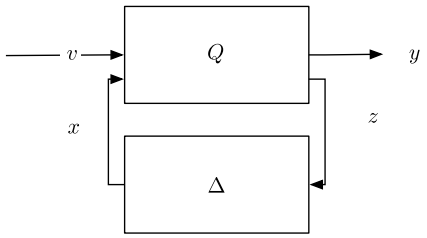


Figure G.5 Compensated plant.

$$||Q_{22}||_\mu < 1 \tag{G.61}$$

$$\Delta - \text{diag}(\Delta_1, ..., \Delta_n) \tag{G.62}$$

$$||\Delta_i|| \leq 1 \tag{G.63}$$

where

$$||Q_{22}||_\mu = \sup\mu(Q_{22}(j\omega)) \tag{G.64}$$

and $\mu(Q)$ is the structured singular value of Q for a block diagonal Δ. Structured uncertainty can be used both to design for performance robustness and to make designs less conservative when designing for stability robustness.

These uncertainty models do not accommodate loop failures and the resulting designs may be unstable when loops or combinations of loops are broken. For a single-input–single-output system, the open-loop performance and stability are just that of the open-loop plant. For a multiinput–multioutput design, the performance will be that of the remainder of the closed-loop plant. With three loops in a 3-axis controlled vehicle, there are three possible systems with one loop broken, and six if two loops break.

G.7.3 Control-structure design

The pairing of inputs and outputs in many systems is of critical importance in designing a low-cost robust system with good performance. Since controls often affect many modes of the system effective pairing can reduce controller complexity.

G.7.4 Nyquist-like techniques

Nyquist-like techniques are extensions of single-input–single-output design for multivariable systems. Three common techniques are sequential loop closing, the characteristic-loci method, and the Nyquist-array approach. These approaches all replace the multivariable problem with a sequence of SISO problems. In each method, the first stage is to decouple the system across a range of frequencies with either a pure gain or dynamic compensator. If the latter is used, there is always the danger of canceling lightly damped poles that often exist in vehicle-control problems. A simple example of

a decoupling stage is in the case of a vehicle with significant off-diagonal inertia terms. The state equations for the system are

$$\dot{x} = \begin{bmatrix} 0 & E \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ I^{-1} \end{bmatrix} u \tag{G.65}$$

where E is a 3×3 identity matrix. If the feedback structure is chosen to be of the form

$$u = IKx \tag{G.66}$$

the state equations become

$$\dot{x} = \begin{bmatrix} 0 & E \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ E \end{bmatrix} Kx \tag{G.67}$$

and K can be designed for the three decoupled loops.

In sequential loop closing, the fastest loops (such as a reaction-wheel tachometer loop) are closed first. If the separation of loop bandwidths is sufficient, this can lead in a straightforward fashion to the final design. When this is not the case, the design usually proceeds in an ad-hoc manner and design decisions made when closing the first loop may have deleterious effects later. The only means available for reducing interaction between the loops is to apply high loop gains.

A second method is a characteristic-locus method that treats the loci of a transfer-function matrix much as one would treat the locus of a single-input–single-output system. With this method, a single-input–single-output controller is designed for each locus.

In the Nyquist-array method, the first step is to introduce a compensator in series with the plant that makes the product of the two diagonally dominant over some range of frequencies. When this is accomplished, the system will behave like a set of SISO loops and the design can proceed.

G.7.5 LQG methods

Linear quadratic Gaussian methods (LQG) are a type of state-space method. The form of an LQG controller is

$$u = -K\hat{x} \tag{G.68}$$

and the estimated state is obtained from

$$\dot{\hat{x}} = (A - BK)\hat{x} + Bu + L(y - C\hat{x}) \tag{G.69}$$

given the state-space model

$$\dot{x} = Ax + Bu + \Gamma w \tag{G.70}$$

$$y = Cx + v \tag{G.71}$$

where w and v are Gaussian white-noise sources for which

$$E(ww^T) = W \tag{G.72}$$

$$E(vv^T) = V \tag{G.73}$$

$$E(wv^T) = N \tag{G.74}$$

The plant matrix A may be of the dynamic system alone, or may include additional dynamics, such as disturbance models, integrators, filters, etc. to add frequency dependence to Eq. (G.68). The gain matrix is one that minimizes the function

$$J = \lim_{T\to\infty} E\left(\int_0^T (z^T Qz + u^T Ru)dt\right) \tag{G.75}$$

where

$$z = Mx \tag{G.76}$$

Q and R can also be functions of frequency.

An LQG controller can exhibit arbitrarily poor robustness properties. Loop-transfer recovery (LTR) was developed to recover the robustness properties of the full-state feedback controller (i.e., one with all states measured). The LTR procedure places some of the estimator eigenvalues at the zeros of the plant, and the remainder is allowed to be arbitrarily fast. This only works if the plant does not have any right-hand-plane zeros near the desired bandwidth and if the plant is square, that is the number of inputs and outputs are equal. If it has unequal numbers of inputs and outputs, LTR can still be applied by squaring the plant. For example, if there are more inputs than outputs, linear combinations of outputs can be added to make the number of outputs equal to the number of inputs. If this is done, recovery can only be applied at the output.

The LQR methodology does not address uncertainty in the plant model directly. Since uncertainty models are not part of the design procedure, it is necessary to test the system over the range of parameters expected in the plant, including higher-frequency dynamics that were neglected in the design. In addition, choosing the weighting matrices Q and R to get the desired performance is not easy, although methodologies for doing so have been suggested, making LQR labor-intensive and expensive to implement.

G.7.6 H_∞ and μ synthesis

The H_∞ (pronounced H-infinity where the H stands for Hardy space) optimal controller is defined as follows. The standard plant representation is shown below. If P is

partitioned into four parts such that

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix} \tag{G.77}$$

the input/output relationship is (see also Fig. G.6)

$$z = [P_{11} + P_{12}K(I - P_{22})^{-1}P_{21}]w \tag{G.78}$$

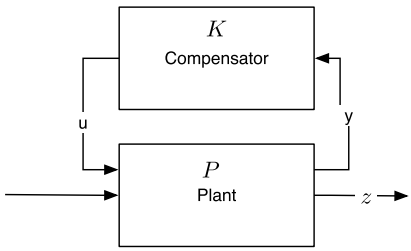


Figure G.6 Standard plant model.

Define

$$F_1(P, K) = [P_{11} + P_{12}K(I - P_{22})^{-1}P_{21}] \tag{G.79}$$

The H_∞ problem is to minimize

$$||F_1(P, k)||_\infty \tag{G.80}$$

over all realizable controllers $K(s)$ that stabilize the closed-loop system. A consistent term for the LQG problem is the H_2 problem since it involves minimizing

$$||F_1(P, k)||_2 \tag{G.81}$$

Many control-design problems such as sensitivity minimization and robustness to perturbations can be formulated as H_∞ problems. However, performance robustness in the face of unmodeled perturbations cannot be formulated as an H_∞ problem. As noted in the above section, the performance robustness is related to the structured singular value of Q. This relationship is used as a basis for the μ-synthesis approach to robust compensator design. The structured singular value cannot be computed directly, and it is necessary to resort to numerical algorithms.

G.8. Single-input–single-output control design

G.8.1 Introduction

The fundamental approach used throughout this chapter is the frequency-domain approach to loop shaping. This is applied both for continuous controllers and discrete-time control systems. All techniques are applied to single-input–single-output control systems in this chapter.

G.8.2 Elementary loop compensation

G.8.2.1 First-order compensators

Control compensators can be created in a number of ways. In this section we will use first-order building blocks to build control systems. The first block of interest is the first-order transfer function

$$K(s) = K\frac{\tau_1 s + 1}{\tau_2 s + 1} \tag{G.82}$$

This is sometimes known as a lead or lag compensator. At low frequencies, it has a gain of K and at high frequencies, its gain is constant and is

$$K(s) = K\frac{\tau_1}{\tau_2} \tag{G.83}$$

It is useful to convert (G.82) into its equivalent phase and magnitude as a function of frequency, where $s = j\omega$. The first step is to make the denominator purely real by multiplying the numerator and the denominator by the complex conjugate of the denominator.

$$K\left(\frac{\tau_1 s + 1}{\tau_2 s + 1}\right)\left(\frac{1 - \tau_1 s}{1 - \tau_2 s}\right) = K\frac{1 + \tau_1\tau_2\omega^2 + j(\tau_1 - \tau_2)\omega}{1 + (\tau_2\omega)^2} \tag{G.84}$$

The phase as a function of frequency is

$$\theta = \frac{(\tau_1 - \tau_2)\omega}{1 + \tau_1\tau_2\omega^2} \tag{G.85}$$

with a peak value of

$$\theta_{peak} = \tan^{-1}\frac{\tau_1 - \tau_2}{2\sqrt{\tau_1\tau_2}} \tag{G.86}$$

at

$$\omega_{peak} = \frac{1}{2\sqrt{\tau_1\tau_2}} \tag{G.87}$$

This compensation element is most useful to add or subtract phase at a desired frequency. Therefore we want to know what the time constants should be. Solving (G.86) and

(G.87) for the time constants gives

$$\tau_2 = \sqrt{\frac{1 + \frac{1}{4}\tan^2\theta_{peak}}{\omega_{peak}^2}} + \frac{\tan\theta_{peak}}{2\omega_{peak}} \tag{G.88}$$

and

$$\tau_1 = \tau_2 + \frac{\tan\theta_{peak}}{\omega_{peak}} \tag{G.89}$$

If the peak phase is negative, the high-frequency gain will be lower than the low-frequency gain. This configuration is often called a low-pass filter or lag compensator. If the peak phase is positive and the high-frequency gain is higher than the low-frequency gain, this is known as a high-pass filter or lead compensator.

G.8.2.2 Generalized integrator

A generalized integrator is

$$K(s) = \frac{\tau s + 1}{s + \sigma} \tag{G.90}$$

If σ and τ are zero this becomes a pure integrator. If only σ is zero, it is an integrator with high-frequency phase compensation. If only τ is zero, it is a low-pass filter with a low-frequency gain of $1/s$.

σ is nonzero when infinite gain at $\omega = 0$ is not necessary. Given a specification that the DC gain equals k, and the phase equal θ_0 (between -90° and 0°) at frequency ω, then the constants are

$$\tau = \frac{\frac{\sigma\tan\theta_0}{\omega} + 1}{\sigma - \omega\tan\theta_0} \tag{G.91}$$

$$\sigma = \frac{1}{k} \tag{G.92}$$

G.8.2.3 Gain compensation of a double-integrator plant

The simplest model of a vehicle is the double integrator that represents one axis of a rigid body,

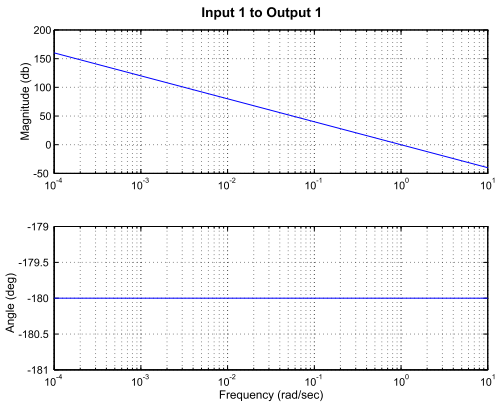
$$I\ddot{\theta} = T \tag{G.93}$$

The output is the attitude and the input is torque. The Laplace transform of Eq. (G.93) is

$$\frac{\Theta(s)}{T(s)} = \frac{1}{Is^2} \tag{G.94}$$

The double-integrator plant is only truly valid if the vehicle is a rigid sphere. Nonetheless, the principles used to compensate it are valid for more complex systems. First, look at the Bode plot of the plant, with an inertia of 1, in Example G.5. The gain is infinite

```
1 a = [0 1;0 0]; b = [0;1];
2 c = [1 0]; d = 0;
3 Fresp(a,b,c,d,1,1,logspace(-4,1))
4 PrintFig(1,1,1,'DoubleIntegrator')
```



Example G.5: Double-integrator model.

at $s = 0$. Therefore any steady input will cause the angle to go to infinity eventually. The response to inputs at other frequencies drops as ω^2. The phase is -180 deg at the gain crossover of $\omega = 1$ rad/s. As a consequence, if we were to close the loop with negative feedback, the plant, if perturbed, would oscillate with a frequency equal to the crossover frequency. Since the phase is -180° everywhere, changing the feedback gain would just shift the frequency of the oscillation. This is clear from adding a $-Kq$ term to the right-hand side of Eq. (G.93)

$$\ddot{\theta} + \frac{K}{I} = \frac{T}{I} \tag{G.95}$$

which is the equation for an oscillator with an oscillation frequency of

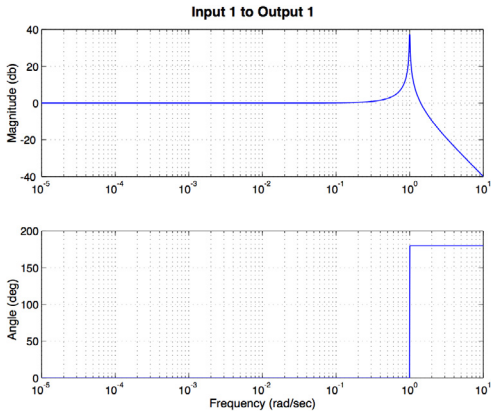
$$\omega = \sqrt{\frac{K}{I}} \tag{G.96}$$

The first question is why bother using feedback at all? Assume that the design requirements state that the only input is a 0.01-N m torque at 0.1 rad/s and that the peak attitude error must be less than 0.001 rad. If the inertia is greater than 1000 kg m^2, the open-loop plant meets specifications and no further work is necessary. If the inertia were too small, we could meet the specification by adding inertia.

The primary reason for using feedback is to deal with uncertainty, or errors, in the models of the plant, the disturbances, and the specifications. The latter should not be easily dismissed. It is not unusual for specifications to change after the design is

done. In this simple example, that might mean a tighter pointing requirement after the vehicle had been built, and increasing its inertia is no longer a viable option. To see how feedback reduces sensitivity to uncertainty, add unity feedback ($K = 1$) to the above vehicle and look at the magnitude plot of the closed-loop system. From $\omega = 0$ to $\omega = 0.2$

```
1 FResp(1,[1 0 1],logspace(-5,1,1000))
```



Example G.6: Torque transmission.

the angle is the same for all frequency inputs. Thus for disturbances at frequencies lower than the resonant frequency, we need not be concerned about their frequency! This is in sharp contrast to the open-loop case where the angle went up as the frequency went down. Thus if the specification were that the response to any single disturbance between 0.001 and 0.4 rad/s was to be less than 0.01 rad, it would be much easier to meet the specification with feedback. At frequencies greater than 2 rad/s, the transfer function between angle and torque is nearly the same as in the open-loop case. This can be seen from the closed-loop transfer function

$$\frac{\Theta(s)}{T(s)} = \frac{1}{Is^2 + K} \tag{G.97}$$

where the first term in the denominator dominates at large frequencies. The range of frequencies for which the control is effective, in this case from 0 to 2 rad/s, is called the bandwidth of the control system. The system is now also much less sensitive to variations in inertia. For all frequencies for which $K \gg Is^2$, the transfer function is

$$\frac{\Theta(s)}{T(s)} = \frac{1}{K} \tag{G.98}$$

Thus the angle response to a disturbance torque is independent of the inertia and depends only on K, the feedback gain. Unfortunately, in this design we have greatly increased our sensitivity to disturbances (or noise) near 1 rad/s due to our control system. Table G.4 summarizes the performance of the closed-loop system.

Table G.4 Control-system performance summary.

Frequency Range	Torque Transmission	Characteristic
$0.01 \leq \omega \leq 0.2$	$\frac{\theta(\omega)}{T(\omega)} = \frac{1}{K}$	Closed loop
$0.02 \leq \omega \leq 0.2$	$\frac{\theta(\omega)}{T(\omega)} = \lvert\frac{1}{K - I\omega^2}\rvert$	Closed loop
$\omega \geq 2$	$\frac{\theta(\omega)}{T(\omega)} = \frac{1}{I\omega^2}$	Open Loop

In summary, this control system has minimal effect on frequencies greater than the bandwidth, which causes the torque to angle transmission to be dominated by the feedback gain at frequencies lower than a decade below the bandwidth, and causes the response to be very sensitive to disturbances or noise at frequencies near the resonant frequency of the system.

In this system, we applied a gain K as the feedback element. This implies that any input, no matter how high its frequency, is amplified by a factor K. In practice, it is not possible to design a controller with a flat response to infinity, but the bandwidth of the sensors and actuators usually exceeds that of the control system, which is usually sufficient.

G.8.2.4 More complex compensation of a double-integrator plant

Table G.5 gives a more interesting specification for our control system in terms of both noise and disturbance.

Table G.5 Control-system specification.

Specification	Frequency Range	Input at any one Frequency	Maximum Angle	Gain
Noise	$\omega \geq 1.0$	0.01 rad	0.0001 rad	-40 db
Disturbance	$\omega = 0$	0.0001 N m	0.0000 rad	∞
Disturbance	$0 < \omega \leq 0.001$	0.0001 N m	0.0001 rad	+120 db

Fig. G.7 gives the block diagram for this system. This is a unity-feedback configuration without any disturbance feedforward and no prefiltering.

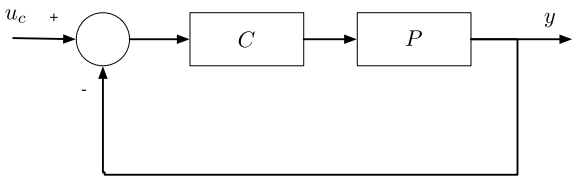


Figure G.7 Unity-feedback block diagram.

The reference is assumed to be 0, and we are only interested in the control system as a regulator. The relationship between the output and the noise and disturbance inputs is

$$\theta = \frac{P}{1+PK}d - \frac{PK}{1+PK}m \tag{G.99}$$

where PK is large, the equation becomes

$$\theta = \frac{1}{K}d - m \tag{G.100}$$

Thus noise is passed into the output with a gain of 1 and the disturbance is attenuated by a factor of K. Where PK is small it becomes

$$\theta = Pd \tag{G.101}$$

which is the open-loop equation.

Eq. (G.100) is a fundamental problem with feedback control. If one applies feedback to attenuate disturbances at a given frequency, any noise at that frequency will be passed into the output.

It is also worth noting that the product of PK is important, not just K alone. Thus the roll-off at high frequencies exhibited by all physical systems means that at high frequencies, PK will go to zero.

The gain specifications are found by substituting the values for the ratios of the output to either the disturbance or noise and computing the product PK. In the case of the disturbance, we must also know P at the frequency of interest. The specifications require -100 dB of roll-off in three decades. For stability purposes, the roll-off must be less than $1/s^2$, or 40 dB/decade at the crossover since for a minimum-phase system a $1/s^2$ slope; this means the phase shift is -180°.

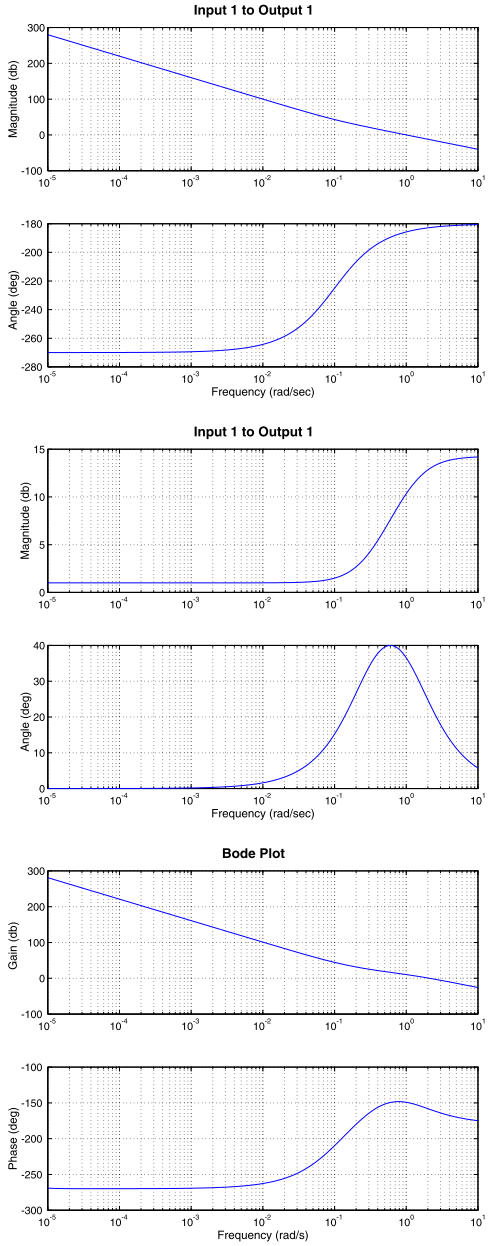
The design of this control system starts by adding an integrator to meet the steady-state offset requirement. This gives the system a gain slope of -60 dB/decade at low frequencies. This must transition to greater than -40 dB/decade at the gain crossover. The form of the integrator is

$$\frac{s+0.1}{s} \tag{G.102}$$

The numerator term removes the phase lag of the integrator at frequencies higher than 0.1 rad/s, thus reducing its impact at the crossover frequency.

At 0.01 rad/s, the integrator only contributes -10° of phase shift to the system. The system remains unstable because, at the crossover (to the right of the graph), the phase will be slightly lower than -180° due to the pole at the origin introduced by the integrator. Our specification calls for a gain of 110 dB at 0.001 rad/s and a gain of -40 dB at 1 rad/s. The difference between those gains is 150 dB. The gain change is correct. However, the plant is unstable.

```
% Plant
a = [0 1;0 0]; b = [0;1];
c = [1 0]; d = 0;
w = logspace(-5,1,1000);

% Integrator
[aI,bI,cI,dI] = ND2SS( [1 0.1],[1 0] );
[a,b,c,d] = Series(a,b,c,d,aI,bI,cI,dI);
FResp(a,b,c,d,1,w)


% Lead
[aL,bL,cL,dL] = LeadLag(0.6,40,1);
FResp(aL,bL,cL,dL,1,w)


% Final system
[a,b,c,d] = Series(a,b,c,d,aL,bL,cL,dL);
[m,p] = FResp(a,b,c,d,1,w,'unwrap');
k = find(p < -270 ); p(k) = p(k) + 360;
yL = {'Gain (db)', 'Phase (deg)'};
h = Plot2D( w, [20*log10(m);p], ...
            'Frequency (rad/s)', ...
            yL, 'Bode Plot', 'xlog');
```



Example G.7: Loop-compensation example.

To correct this, we would like to adjust the phase at the crossover without changing the gain much. This can be done with lead compensation. We need to add about +60° of phase shift at 0.4 rad/s.

The use of standard control elements, leads, lags, and PI can be used to achieve the desired performance. This procedure requires iteration on the design. After loop compensation, the control loop needs to be tested to make certain that sufficient control authority is available for the system.

G.9. Digital control

G.9.1 Introduction

Vehicle control systems are usually implemented on digital computers. This section discusses digital control-system design techniques.

G.9.2 Modified continuous design

G.9.2.1 Introduction

Most control engineers are first taught the control of continuous systems in which the plant to be controlled is described by differential equations

$$\dot{x} = f(x, u, t) \tag{G.103}$$

and the measurements are described by the nonlinear relationship

$$y = h(x, u, t) \tag{G.104}$$

The linearized version of these equations are

$$\dot{x} = Ax + Bu \tag{G.105}$$

and

$$y = Cx + Du \tag{G.106}$$

Corresponding to these equations is the frequency-domain representation of these equations

$$y(s) = (C(sE - A)^{-1}B + D)u(s) \tag{G.107}$$

Fig. G.8 shows a block diagram of a control system and plant. The only discrete part of the system is the computer and its software, the remainder of the system is continuous. Measurements are periodically entered into the computer using a sampling mechanism and numbers representing the desired actuation are converted to continuous operations using the discrete-to-continuous conversion block.

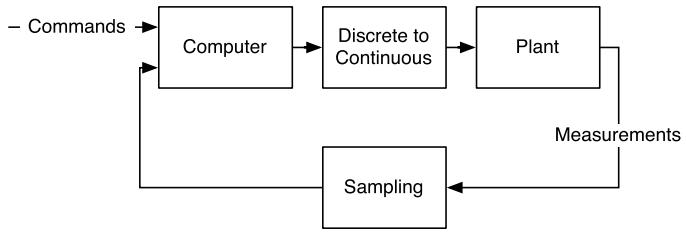


Figure G.8 Computer-controlled system.

The discrete-to-continuous conversion block may take several forms. The most common is the sample and hold. During every control period, an external register is set to the number representing the desired actuation. This value is maintained for the entire control period. For example, a register that is an input to a digital-to-analog converter that determines the input voltage to a reaction wheel might be set to the desired value. A single bit that drives a thruster-valve relay might be set and held high for the duration of the command period. Thrusters are examples of actuators that usually have only two states; these are sometimes driven by pulsewidth modulation. The output of the control system, which is a torque level for that output, is converted into a pulsewidth. If the desired torque level is u, and the control period is T, then the pulsewidth is

$$\tau = \frac{uT}{u_c} \tag{G.108}$$

where u_c is the torque available from the actuator. Thus the average torque over the control period equals u. The pulsewidth will be limited by the resolution of the timing mechanism, and the speed of response of the actuator.

Since the output sampling rate does not need to be fixed, it is also possible to vary the actuation period on the output. This is known as pulsewidth–pulse-frequency modulation.

The modified frequency continuous design approach represents the conversion of the measurements to digital form, and the conversion of the commands to continuous from a continuous function. This permits the use of frequency-domain design techniques and the nearly direct conversion of the resulting algorithms into discrete form, with minimal effort. Since many modern control-design techniques are continuous-time techniques, this permits a wider variety of design approaches.

In this section, we will look at the key elements that give a continuous-time system the characteristics of a discrete-time system. These are the sampler, the delay, and the discrete-to-continuous conversion function. For the latter, we will look at the sample and hold and pulsewidth modulation.

G.9.2.2 The sampler

The sampler is represented in the continuous-time domain as a gain of $1/T$, where T is the sample period.

G.9.2.3 The delay

All digital control systems have true delays, as opposed to lags. A delay means that a signal, regardless of frequency, is always the delay time late. The Laplace transform of a delay is

$$e^{-s\Delta} \tag{G.109}$$

where Δ is the delay in seconds and s is $j\omega$. This is a nonminimum-phase function. Nonminimum phase means that the phase angle is greater than the minimum possible for a function with the same magnitude. Eq. (G.109) can be expanded as

$$e^{-s\Delta} = \cos\omega\Delta - j\sin\omega\Delta \tag{G.110}$$

The magnitude of this function is 1 and the phase is $-\omega\Delta$. Thus the phase decreases linearly with frequency. This is also known as an all-pass function. The minimum phase for a constant magnitude function is 0. Therefore this function is a nonminimum phase. All-pass functions are always nonminimum phase.

Any uncertainty in a delay can wreak havoc on control approaches that require precise phase knowledge at high frequencies since the phase error will be proportional to the frequency. For example, if one has a >20% delay uncertainty, in a 0.3-second delay, then the phase uncertainty at 10 rad/s, for example, will be >34°. Eq. (G.109) is difficult to use directly in s-plane analysis. Instead, series expansions are employed to represent it. The Taylor-series expansion for an exponential is

$$e^{-sa} = 1 + sa + \frac{(sa)^2}{2} + \frac{(sa)^3}{6} + \frac{(sa)^4}{24} + \frac{(sa)^5}{120} + \ldots + \frac{(sa)^n}{n!} \tag{G.111}$$

The transfer function of the exponential should be proper, that is to say the denominator order must be greater than or equal to the order of the numerator. The general approach to expanding Eq. (G.109) is to write it as a fraction

$$e^{-s\Delta} = \frac{e^{-sf\Delta}}{e^{s(1+f)\Delta}} \tag{G.112}$$

where f is less than 1. If f is zero, for example, the first-order expansion is

$$e^{-s\Delta} = \frac{1}{1+s\Delta} \tag{G.113}$$

This function will represent the exponential at very low frequencies, but does not capture its minimum-phase nature. A widely used representation is with $f = 1/2$. To second order this is

$$e^{-s\Delta} = \frac{1 - \frac{s\Delta}{2} + \frac{(s\Delta)^2}{8}}{1 + \frac{s\Delta}{2} + \frac{(s\Delta)^2}{8}} \tag{G.114}$$

Note that this function has a pair of zeros in the right half-plane and is all-pass. If f and $1-f$ are not equal the expansions in the numerator and denominator should be carried out so that their relative error is equal. For example, if $f = 1/3$ the appropriate function is

$$e^{-s\Delta} = \frac{1 - \frac{s\Delta}{3}}{1 + \frac{2s\Delta}{3} + \frac{(s\Delta)^2}{6}} \tag{G.115}$$

and if $f = 2/5$

$$e^{-s\Delta} = \frac{1 - \frac{2s\Delta}{5} + \frac{(s\Delta)^2}{20}}{1 + \frac{3s\Delta}{5} + \frac{3(s\Delta)^2}{20} + \frac{(s\Delta)^3}{60}} \tag{G.116}$$

where (G.114), (G.115), and Eq. (G.116) are known as Padé approximants.

Example G.8 shows (G.113) (the solid line) and (G.114) (the dot-dashed line) expanded only to first order with $D = 0$ and compared with the exponential (the dashed line). (G.113) rolls off at high frequency and does not match the phase well above 0.2 rad/s. The nonminimum phase-transfer function matches the magnitude exactly and the phase to about 2 rad/s. For frequencies less than $0.1/D$, the delay has minimal effect and can be ignored. Assume that we are designing a rate-control system whose plant is a single integrator with a delay. Represent the delay by (G.114) expanded only to second order. A gain is a perfectly adequate controller for a single integrator. The plant is

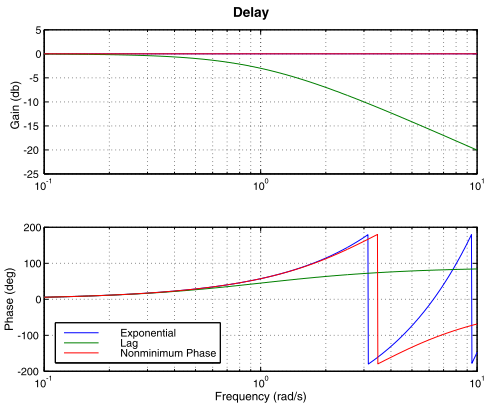
$$P(s) = \frac{1 - \frac{s\Delta}{2}}{s(1 + \frac{s\Delta}{2})} \tag{G.117}$$

The controller is K and the closed-loop transfer function is

$$\frac{y}{r} = \frac{K(1 - \frac{s\Delta}{2})}{K + \frac{\Delta}{2}s^2 + \left(1 - K\frac{\Delta}{2}\right)s} \tag{G.118}$$

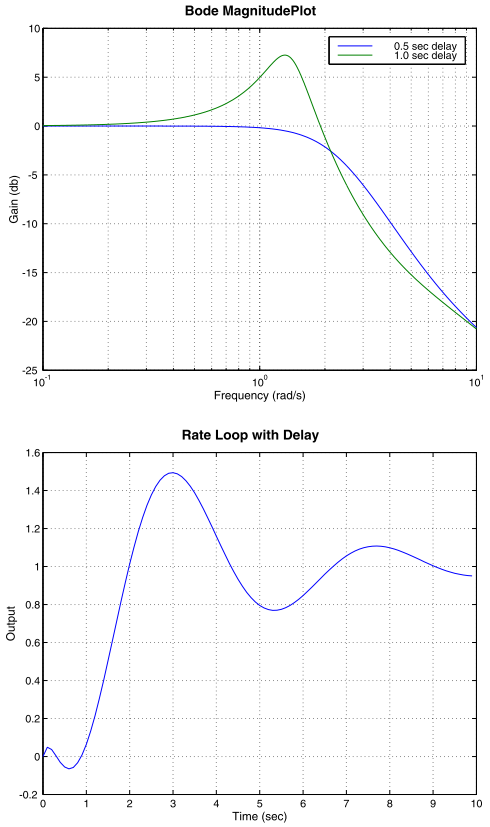
If $D = 0$ this reduces to the standard equation for rate feedback. The system will become unstable if K exceeds $2/D$. Thus the delay limits the system gain, a natural consequence of a right half-plane zero. Example G.9 shows the Bode magnitude plot with $K = 1$ and $D = 1$. The solid line shows the magnitude response with $D = 0.5$ and the dashed line shows it with $D = 1.0$. The damping decreases as the delay increases. The shape of the magnitude curve is also somewhat different from what you would expect for a

```
% Plant
clear angle
a = [0 1;0 0]; b = [0;1];
c = [1 0]; d = 0;
n       = 1000;
w       = logspace(-1,1,n);
tau     = 1;
g       = exp(1i*w);
p       = zeros(3,n);
m       = zeros(3,n);
m(1,:) = abs(g);
p(1,:) = (180/pi)*angle(g);
[num,den] = Pade( 0, 1, tau );
[m(2,:),p(2,:)] = FResp(num,den,w);
[num,den] = Pade( 2, 2, tau );
[m(3,:),p(3,:)] = FResp(num,den,w);
yL = {'Gain (db)', 'Phase (deg)'};
h = Plot2D( w, [20*log10(m);p], 'Frequency (rad/s)',...
            yL, 'Delay', 'xlog', {'[1:3]' '[4:6]'});
legend('Exponential', 'Lag', 'Nonminimum Phase')
set(h,'color',[1 1 1]);
```



Example G.8: Delay example.

```
% Plant
aP = 0; bP = 1;
cP = 1; dP = 0;
w       = logspace(-1,1,1000);
[aD, bD, cD, dD] = Delay(0.5,2);
[a,b,c,d] = Series(aD, bD, cD, dD, aP, bP, cP, dP);
[m1,p]    = FResp(a-b*c,b,c,d,1,1,w);
[aD, bD, cD, dD] = Delay(1.0,2);
[a,b,c,d] = Series(aD, bD, cD, dD, aP, bP, cP, dP);
[m2,p]    = FResp(a-b*c,b,c,d,1,1,w);
yL = 'Gain (db)';
h = Plot2D( w, 20*log10([m1;m2]), 'Frequency (rad/s)',...
            yL, 'Bode MagnitudePlot', 'xlog');
legend('0.5 sec delay','1.0 sec delay')
g = statespace(a-b*c,b,c,d,'Rate Control');
[y,x,t] = Step(g,1,0.1,100);
Plot2D(t,y,'Time (sec)', 'Output', 'Rate Loop with Delay')
```



Example G.9: Bode magnitude plot of the rate-control loop with delays of 0.5 and 1 second.

second-order system with the same damping ratio. At high frequencies, it appears to be concave instead of convex.

The zero has an interesting effect on the transient response. At low frequencies, the transfer function is 1. At high frequencies, the transfer function is

$$\frac{y}{r} = -\frac{K}{s} \tag{G.119}$$

The sign is reversed! Therefore any input (such as a step) with sufficiently high-frequency content will cause the output to go in the opposite direction of the input at first. This is illustrated in the bottom plot of Example G.9. This effect can be reduced by shortening the delay or low-pass filtering the command input.

G.9.2.4 The zero-order hold

The zero-order hold is the standard method of driving analog actuators with signals from computers. The zero-order hold is given by

$$u(t) = u_{-1}(t) - u_{-1}(t - T) \tag{G.120}$$

This is just the difference of a step and a delayed step. The Laplace transform of Eq. (G.113) is

$$u(s) = \frac{1 - e^{-sT}}{s} \tag{G.121}$$

When multiplied by the gain of the sampler, $1/T$, it becomes

$$u(s) = \frac{1 - e^{-sT}}{sT} \tag{G.122}$$

Eq. (G.122), when included in the continuous-time system, represents the effects of sampling on input and the zero-order hold on output. This allows us to include the effects of the digital system while designing in the continuous-time domain. Eq. (G.122) is most easily handled using its series expansion. This is done by substituting Eq. (G.115) or Eq. (G.116) for the delay in the numerator. The result is

$$u(s) = \frac{1 - \frac{sT}{6}}{1 + \frac{sT}{3}} \tag{G.123}$$

for Eq. (G.115) and

$$u(s) = \frac{1 - \frac{sT}{10} + \frac{(sT)^2}{60}}{1 + \frac{2sT}{5} + \frac{(sT)^2}{20}} \tag{G.124}$$

for Eq. (G.116). Substituting Eq. (G.114), would not have provided the right half-plane zeros that give the zero-order hold its nonminimum phase character.

G.9.2.5 Pulsewidth modulation

Pulsewidth modulation involves varying the pulsewidth from 0 to T. While this cannot be directly represented, as can the zero-order hold, the effect can be analyzed by assuming that the pulsewidth is fixed at t, a fraction of T. The pulsewidth modulator is

$$u(t) = u_{-1}(t - T + \tau) - u_{-1}(t - T) \tag{G.125}$$

The Laplace transform (including the sampler gain) is

$$u(s) = \frac{e^{-s(T-\tau)} - e^{-sT}}{sT} \tag{G.126}$$

or, after some manipulation

$$u(s) = \left(\frac{e^{s\tau} - 1}{e^{sT} - 1}\right)\left(\frac{1 - e^{sT}}{sT}\right) \tag{G.127}$$

The first term in parentheses is an additional nonminimum-phase function. Using

$$e^{sT} - 1 = \frac{1 + \frac{sT}{10} + \frac{(sT)^2}{60}}{1 - \frac{2sT}{5} + \frac{(sT)^2}{20}}\frac{1}{s} \tag{G.128}$$

and substituting gives

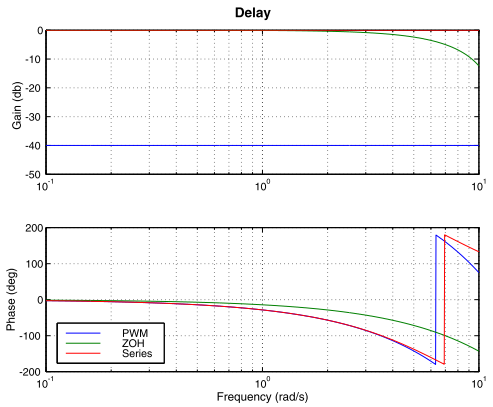
$$\frac{e^{s\tau} - 1}{e^{sT} - 1} = \frac{\tau}{T}\left(\frac{1 + \frac{s\tau}{10} + \frac{(s\tau)^2}{60}}{1 - \frac{2s\tau}{5} + \frac{(s\tau)^2}{20}}\right)\left(\frac{1 - \frac{2sT}{5} + \frac{(sT)^2}{20}}{1 + \frac{sT}{5} + \frac{(sT)^2}{60}}\right) \tag{G.129}$$

The finite pulsewidth introduces a very high-frequency zero pair and an unstable pole pair. For frequencies below the sampling frequency, determined by T, this transfer function does not appreciably affect the dynamics. Therefore the transfer function for the pulsewidth modulator is

$$u(s) = \frac{\tau}{T}\left(\frac{11\frac{2sT}{5} + \frac{(sT)^2}{20}}{1 + \frac{sT}{5} + \frac{(sT)^2}{60}}\right)\left(\frac{1 - \frac{sT}{10} + \frac{(sT)^2}{60}}{1 + \frac{2sT}{5} + \frac{(sT)^2}{20}}\right) \tag{G.130}$$

Example G.10 shows the Bode plot for the exact and series expansion, as well as the zero-order hold, for $T = 0.5$ and $t = 0.005$. The solid line is the series expansion for the pulsewidth modulator, the dashed line is the exact representation, and the dot-dashed line is the zero-order hold. The gain on the zero-order hold was made identical to that of the pulsewidth modulator for comparison purposes only. The pulsewidth modulator has a greater phase shift than the zero-order hold and no high-frequency roll-off. It will tend to interact more with high-frequency dynamics.

```
w       = logspace(-1,1,1000);
tau     = 0.005;
T       = 0.5;
s       = j*w;
% PWM
g       = (exp(-s*(T-tau)) - exp(-s*T))./(s*T);
m(1,:) = abs(g);
p(1,:) = (180/pi)*angle(g);
% ZOH
g       = (1-exp(-s*T))./(s*T);
m(2,:) = abs(g);
p(2,:) = (180/pi)*angle(g);
[num,den] = PWM( tau, T, 2 );
[m(3,:),p(3,:)] = FResp(num,den,w);
yL = {'Gain (db)', 'Phase (deg)'};
h = Plot2D( w, [20*log10(m);p], ...
            'Frequency (rad/s)', ...
            yL, 'Delay', 'xlog', ...
            {'[1:3]' '[4:6]'});
legend('PWM', 'ZOH', 'Series')
set(h,'color',[1 1 1]);
```



Example G.10: Comparison of pulsewidth modulator and zero-order hold.

G.10. Continuous-to-discrete transformations

G.10.1 The difference equation

Once we have a continuous design, it is necessary to convert it into a set of difference equations for implementation in the computer. The controller will be of the form

$$\zeta_i = \sum_{j=i-1}^{i-n} a_j x_j + \sum_{j=i}^{i-m} b_j u_j \tag{G.131}$$

where u_j is the jth input and x_j is the jth output. The time between inputs and outputs is the sample period T. If we define z as a one sample-period delay, we can transform the difference equation into

$$x(z) = \sum_{j=1}^{n} a_j z^j x(z) + \sum_{j=0}^{m} b_j z^j u(z) \tag{G.132}$$

For example, the difference equation for a trapezoidal rule integrator is

$$x_i = x_{i-1} + \frac{T}{2}(u_i + u_{i-1}) \tag{G.133}$$

When it has $n = 1$ and $m = 1$ and is transformed into the z domain it is

$$\frac{x(z)}{u(z)} = \frac{T}{2}\frac{z+1}{z-1} \tag{G.134}$$

G.10.2 Transforming from the s plane to the z plane

The delay z in the s plane is

$$z = e^{sT} \tag{G.135}$$

and

$$s = \frac{\log z}{T} \tag{G.136}$$

Eq. (G.135) and Eq. (G.136) allow us to move between the s and z planes. Given a continuous time-transfer function, the equivalent s domain function can be found by substituting Eq. (G.136) for s. Since the natural logarithm does not easily convert to difference equations, we expand it in a series.

$$s = \frac{2}{T}\left(\left(\frac{z-1}{z+1}\right) + \frac{1}{3}\left(\frac{z-1}{z+1}\right)^3 + \ldots + \frac{1}{n}\left(\frac{z-1}{z+1}\right)^n\right) \tag{G.137}$$

where n is odd. The first-order expansion is known as the Tustin transformation and is

$$s = \frac{2}{T}\left(\frac{z-1}{z+1}\right) \tag{G.138}$$

The second-order expansion is

$$s = \frac{8}{3T}\left(\frac{z^3-1}{(z+1)^3}\right) \tag{G.139}$$

Since s is the derivative operator, one could also replace it with a first difference

$$s = \frac{1}{T}(z-1) \tag{G.140}$$

or a second difference

$$s = \frac{1}{T}(3z^2 - 4z + 1) \tag{G.141}$$

G.10.3 Transformation of a differentiator

A differentiator with roll-off at high frequencies is

$$\frac{s}{\tau s + 1} \tag{G.142}$$

The response to a ramp at $t = \infty$ can be found by the final-value theorem

$$\lim_{t\to\infty} = \lim_{s\to 0} s\left(\frac{s}{\tau s+1}\right)\frac{1}{s^2} = 1 \tag{G.143}$$

This can be implemented with a first difference. The result is

$$\frac{1}{\tau}\left(\frac{z-1}{z-\left(1-\frac{T}{\tau}\right)}\right) \tag{G.144}$$

where $\tau > T$. The final-value theorem can be applied to this function

$$\lim_{t\to\infty} = \lim_{s\frac{1}{\tau}\to 0} s\left(\frac{e^{sT}-1}{e^{sT}-\left(1-\frac{T}{\tau}\right)}\right)\frac{1}{s^2} = \frac{1}{T}\frac{e^{sT}-1}{s} = \frac{sT}{sT} = 1 \tag{G.145}$$

The matched pole-zero method is also easy to apply. Transforming the zero and pole gives

$$K\frac{z-1}{z-e^{T/\tau}} \tag{G.146}$$

The gain is found using the final-value theorem

$$K\lim_{s\to 0}\left(\frac{1}{e^{sT}-e^{T/\tau}}\right)\frac{e^{sT}-1}{s} = 1 \tag{G.147}$$

or

$$K = \frac{1-e^{T/\tau}}{T} \tag{G.148}$$

These results, along with the Tustin and the 2nd difference are summarized in Table G.6.

All four rate-estimator errors are shown below.

G.10.4 State estimator

An alternative is to use a state estimator. The discrete-time state equations for a double-integrator plant are

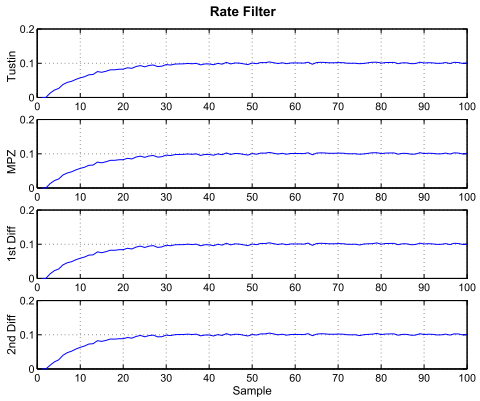
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}_{k+1}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_k + \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix}u + \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}\left(y - \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_k\right) \tag{G.149}$$

The first state is the estimated angle and the second is the estimated rate. y is the measured angle. Expanding the state equations as difference equations gives

$$zx_1 = x_1 + Tx_2 + k_1 - k_1x_1 \tag{G.150}$$

$$zx_2 = x_2 + k_2y - k_2x_1 \tag{G.151}$$

```
T         = 0.1;
tau       = 1;
% Tustin
[nT, dT] = S2Z( [1 0], [tau 1], T, ...
                'bilinear' );
% MPZ
k         = (1 - exp(-T/tau))/T;
nM        = k*[1 -1];
dM        = [1 -exp(-T/tau)];
% First difference
nF        = (1/tau)*[1 -1];
dF        = [1 -(1-T/tau)];
% 2nd difference
nS        = [3 -4 1]/tau/3;
dS        = [3 -4 1 + 2*T/tau]/3;
% Sim
nSim      = 100;
y         = zeros(4,nSim);
u         = linspace(0,1,nSim) + 0.002*randn(1,
        nSim);
for k = 3:nSim
  y(1,k) = -dT(2)*y(1,k-1) + nT(1)*u(k) ...
             + nT(2)*u(k-1);
  y(2,k) = -dM(2)*y(2,k-1) + nM(1)*u(k) ...
             + nM(2)*u(k-1);
  y(3,k) = -dF(2)*y(3,k-1) + nF(1)*u(k) ...
             + nF(2)*u(k-1);
  y(4,k) = -dS(2)*y(4,k-1) ...
           - dS(3)*y(4,k-2) ...
           + nS(1)*u(k) ...
           + nS(2)*u(k-1) ...
           + nS(3)*u(k-2);
end
yL = {'Tustin' 'MPZ' '1st Diff' '2nd Diff'};
h = Plot2D( 1:nSim, y, 'Sample',yL, 'Rate Filter'
        );
set(h,'color',[1 1 1]);
PrintFig(0,1,1,'RateEst')
err = mean(y(:,(nSim/2):end) - 0.1,2)
%err =
%   0.00096175387563
%   0.00096088706325
%   0.00100900220493
%   0.00110102280894
```



Example G.11: Comparison of the four rate estimators.

Solving the first equations for x_1 gives

$$x_1 = \frac{Tx_2 + k_1 y}{z - 1 + k_1} \tag{G.152}$$

and substituting into the second equation for x_1 gives the transfer function

$$\frac{x_2(z)}{y(z)} = k_2 \frac{z - 1}{z^2 + (k_1 - 2)z + Tk_2 - k_1 + 1} \tag{G.153}$$

which has a second-order denominator. The gains can be chosen in many ways. One is just to map the desired s-plane poles into the z-plane using the relationship (see Table G.6)

$$z = e^{sT} \tag{G.154}$$

Table G.6 Z-plane representation of a filtered differentiator.

Form	Transfer Function
First Difference	$\frac{1}{\tau}\left(\frac{z-1}{z-\left(1-\frac{T}{\tau}\right)}\right)$
Second Difference	$\frac{1}{\tau}\left(\frac{3z^{-2}-4z+1}{3z^2-4z+1+\frac{2T}{\tau}}\right)$
Tustin	$\frac{2/\tau}{1+\frac{T}{2\tau}}\frac{z-1}{z-\left(\frac{1-\frac{T}{2\tau}}{1+\frac{T}{2\tau}}\right)}$
Matched Pole Zero	$\frac{1-e^{-T/\tau}}{T}\left(\frac{z-1}{z-e^{-T/\tau}}\right)$
State Estimator	$\frac{x_2(z)}{y(z)}=k_2\frac{z-1}{z^2+(k_1-2)z+Tk_2-k_1+1}$

where

$$s=-\zeta\omega\pm j\omega\sqrt{1-\zeta^2} \tag{G.155}$$

so

$$z=e^{-\zeta\omega T}e^{\pm j\theta} \tag{G.156}$$

Note that

$$e^{j\theta}=\cos\theta+j\sin\theta \tag{G.157}$$

so the roots are

$$z=e^{-\zeta\omega T}\left[\begin{array}{cc}\cos\theta+j\sin\theta & \cos\theta-j\sin\theta\end{array}\right] \tag{G.158}$$

This gives

$$\theta=\omega T\sqrt{1-\zeta^2} \tag{G.159}$$

$$k_1=2(e^{-\zeta\omega T}\cos\theta+1) \tag{G.160}$$

$$k_2=\frac{e^{-\zeta\omega T}(1+2\cos\theta)+1}{T} \tag{G.161}$$

where

$$\theta=\omega\sqrt{1-\zeta^2} \tag{G.162}$$

The roots of the transfer function in z are

$$z = -\frac{k_1 - 2}{2} \pm \sqrt{\frac{(k_1-2)^2}{4} - Tk_2 + k_1 - 1} \tag{G.163}$$

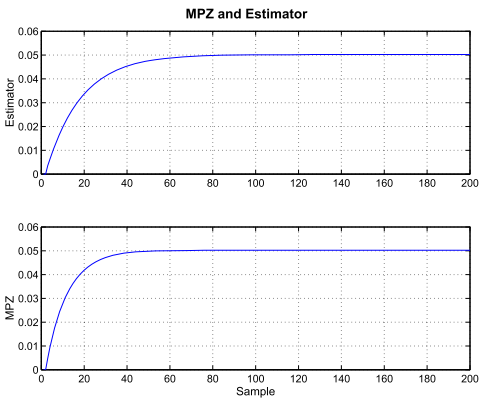
Solving for k_1 and k_2

$$k_1 = 2\left(1 - e^{-\zeta\omega T}\cos\theta\right) \tag{G.164}$$

$$k_2 = \frac{1}{T}\left(\frac{(k_1-2)^2}{4} + k_1 - 1 - \sin^2\theta e^{-2\zeta\omega T}\right) \tag{G.165}$$

The s-plane parameters for the estimator are $z = 0.7072$ and $\omega = 10.0$.

```
T         = 0.1;
tau       = 1;
zeta      = 0.7071;
omega     = 10.0;
% MPZ
k         = (1 - exp(-T/tau))/T;
nM        = k*[1  -1];
dM        = [1 -exp(-T/tau)];
% Estimator
a         = [1 T;0  1];
theta     = omega*T*sqrt(1-zeta^2);
c         = cos(theta);
e         = exp(-zeta*omega*T);
k1        = 2*(1 - e*c);
s         = sin(theta);
k2        = ((k1-2)^2/4 + k1 - 1 - e*s)/T;
K         = [k1;k2];
c1        = (K(1)-2);
c0        = T*K(2)-K(1)+1;
% Sim
nSim = 200;
y = zeros(2,nSim);
u = linspace(0,1,nSim) + 0.00*randn(1,nSim);
x = zeros(2,1);
for k = 3:nSim
  y(1,k) = K(2)*(u(k) - u(k-1)) ...
             - c1*y(1,k-1) - c0*y(1,k-2);
  %x       = a*x + K*(u(k)-x(1));
  y(2,k) = -dM(2)*y(2,k-1) + nM(1)*u(k) ...
             + nM(2)*u(k-1);
end
yL = {'Estimator' 'MPZ'};
h = Plot2D( 1:nSim, y, 'Sample',yL, ...
            'MPZ and Estimator');
set(h,'color',[1 1 1]);
PrintFig(0,1,1,'MPZandEstimator')
```



Example G.12: Comparison of the matched pole zero with the estimator.

The Tustin and first difference are just low-order expansions of the matched pole zero. The second difference has zeros at 1 and 1/3 and poles at

$$\frac{4 \pm \sqrt{4 - \frac{24T}{\tau}}}{6} \tag{G.166}$$

G.11. Flexible-structure control

G.11.1 Introduction

Most vehicles cannot be considered rigid bodies when their control systems are being designed. Generally, a flexible body is one in which the structure deforms and those deformations are much smaller than the rigid-body motion. However, many other types of dynamics can be included in this category.

G.11.2 Two coupled inertias

The simplest possible flexible-body problem is that of two coupled inertias, illustrated in Fig. G.9. The inertias are connected by a torsional spring and damper. A control torque can be applied to body 1. Disturbances may be applied to either body. The linear differential equations for the coupled set are

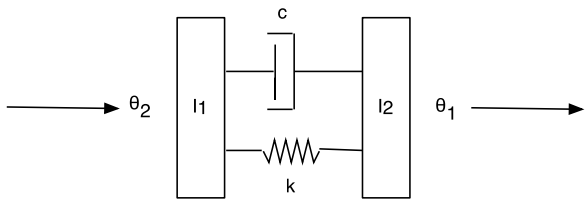


Figure G.9 Two inertias coupled by a spring and a damper.

$$I_1\ddot{\theta}_1 + c(\dot{\theta}_1 - \dot{\theta}_2) + k(\theta_1 - \theta_2) = u + d_1 \tag{G.167}$$

$$I_2\ddot{\theta}_1 + c(\dot{\theta}_2 - \dot{\theta}_1) + k(\theta_2 - \theta_1) = d_2 \tag{G.168}$$

Taking the Laplace transform of the equations gives

$$\begin{bmatrix} I_1 s^2 + cs + k & -cs - k \\ -cs - k & I_2 s^2 + cs + k \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} u + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{G.169}$$

and solving for θ_1 and θ_2 gives

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \frac{\begin{bmatrix} I_1 s^2 + cs + k & -cs - k \\ -cs - k & I_2 s^2 + cs + k \end{bmatrix}}{I_1 I_2 s^2 (s^2 + c\frac{\gamma}{I_2} s + k\frac{\gamma}{I_2})} \begin{bmatrix} 1 \\ 0 \end{bmatrix} u \tag{G.170}$$

where γ is the modal coupling factor and is

$$\gamma = \frac{I_1 + I_2}{I_1} \tag{G.171}$$

and γ can range from 1 to ∞.

G.11.3 Double integrator

The collocated plant reduces to a double integrator if $\gamma \to 1$. The double integrator can be controlled by a simple lead network.

$$C(s) = K\frac{\tau_1 s + 1}{\tau_2 s + 1} \tag{G.172}$$

where $\tau_1 > \tau_2$. That is, the zero is at a lower frequency than the pole. This can also be thought of as a proportional-derivative controller cascaded with a low-pass filter. The advantage of using the pole/zero combination is that the phase-lag effects of the pole are automatically accounted for in the design.

First, choose the crossover frequency and create a lead network that provides a + phase shift at that frequency and then adjust the gain so the crossover (where the gain is 1) is at that frequency. This is shown in the following example. The crossover is set to 0.05 rad/s. 60° of lead is selected. In the first example the damping ratios for the complex modes are low. When the gain is lowered by 63 dB (so now the crossover is centered in the lead peak) the damping ratios are all 1.

G.11.4 Control algorithms

Two cases are of interest. The first is the collocated sensor and actuator case. The transfer function from u to θ_1 is

$$\frac{\theta_1}{u} = \frac{s^2 + \frac{c}{I_2}s + \frac{k}{I_2}}{I_1 s^2(s^2 + c\frac{\gamma}{I_2}s + k\frac{\gamma}{I_2})} \tag{G.173}$$

When $I_1 \gg I_2$ the modal coupling factor is 1 and the quadratic factor in the denominator equals the quadratic factor in the numerator and they drop out. The resulting transfer function is for a rigid body with inertia I_1.

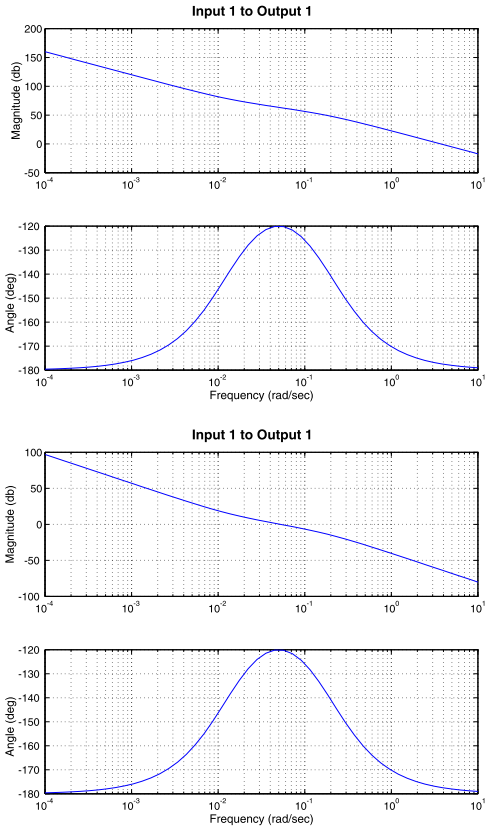
At frequencies much lower than the roots of the denominator quadratic factor, the first two terms in the quadratic factor are negligible and the transfer function reduces to a rigid body with inertia $I_1 + I_2$. At frequencies much higher than the complex poles, the transfer function reduces to a rigid body with inertia I_1.

Since γ is always greater than 1, the pole pair determined by the quadratic term in the denominator will always be at a higher frequency than the zeros. Therefore the phase that starts at -180 deg due to the double pole at the origin, will flip to 0° and then return to -180°. No matter how many flexible-body modes there are, if the sensor and actuator are collocated, the phase is never less than -180°.

The damping ratio for the fixed-base mode is

$$\zeta = \frac{c}{s\sqrt{kI_2}} \tag{G.174}$$

```
% Double integrator
[aP, bP, cP, dP] = ND2SS( 1, [1 0 0] );
% Lead
[aL, bL, cL, dL] = LeadLag( 0.05, 60, 0 );
[a,b,c,d] = Series(aL,bL,cL,dL,aP,bP,cP,dP);
FResp(a,b,c,d);
s    = eig(a-b*c)
zeta = S2Damp(s)
% s =
%
%    -0.0134
%    -0.0866 + 3.7307i
%    -0.0866 - 3.7307i
%
%
% zeta =
%
%      1.0000
%      0.0232
%      0.0232
% With gain adjustment
[aL, bL, cL, dL] = LeadLag( 0.05, 60, -63 );
[a,b,c,d] = Series(aL,bL,cL,dL,aP,bP,cP,dP);
FResp(a,b,c,d);
s    = eig(a-b*c)
zeta = S2Damp(s)
```



Example G.13: Control of a double integrator with a lead network.

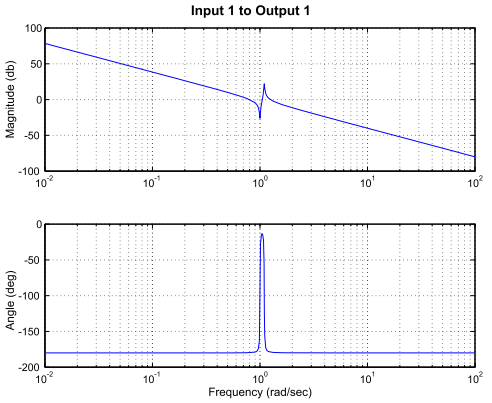
and the damping ratio for the moving-base case is

$$\zeta = \frac{c}{s\sqrt{\gamma k I_2}} \tag{G.175}$$

Since γ is always greater than one, the damping ratio will always be less than that measured on the ground. The poles and zeros of the quadratic terms are always complex since flexible modes (if they are of concern at all) are lightly damped. The numerator zeros are the fixed-base frequencies of the flexible part of the system (represented by I_2 and its spring and damper). They are the frequencies that would be measured if the flexible part were cantilevered to a fixed base. It is easy to go from the fixed-base vibrational frequencies to the frequencies of the vehicle if one knows the inertias.

The Bode plot for this transfer function for $I_1 = 1$, $I_2 = 0.1$, fixed base $\zeta = 0.005$ and fixed-base frequency equal to 1 rad/s is illustrated in Example G.14.

```
I1     = 1;
I2     = 0.2;
zeta   = 0.005;
omega  = 1;
k      = I2*omega^2;
gamma  = (I1 + I2)/I1;
c      = 2*zeta*omega*I2;
num    = [1 c/I2 k/I2];
den    = [I1 I1*c*gamma/I2 I1*k*gamma/I2 0 0];
FResp(num,den)
PrintFig(1,1,1,'Collocated')
```



Example G.14: Bode plot for the collocated sensor and actuator transfer function.

Note that this system has multiple crossovers. This is a characteristic of flexible-body systems that is not seen in many other control problems.

Suppose that instead of measuring θ_1, we measure θ_2. θ_2 is separated from the actuator by the flexible coupling. The transfer function is

$$\frac{\theta_2}{u} = \frac{\frac{c}{I_2}s + \frac{k}{I_2}}{I_1 s^2(s^2 + c\frac{\gamma}{I_2}s + k\frac{\gamma}{I_2})} \tag{G.176}$$

The numerator is a single real zero at frequency k/c. In terms of damping ratio this is

$$s = \frac{\omega}{2\zeta} \tag{G.177}$$

If ζ is small, this is much higher than the denominator frequency. Therefore in the noncollocated system, the phase will decrease to -360° after the pole pair and will not return to -180° until a much higher frequency.

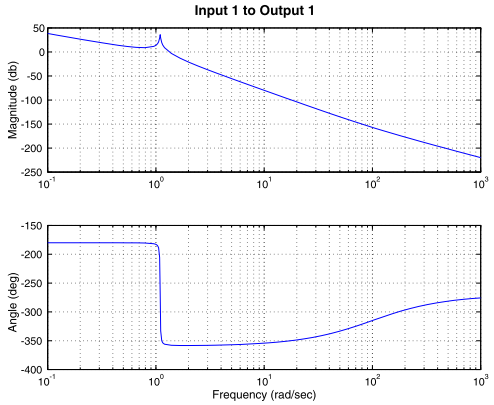
Note that the system behavior at very high and very low frequencies (relative to the complex) pole is not changed. It is only the behavior at intermediate frequencies that is different.

The Bode plot for the noncollocated transfer function is illustrated in Example G.15. The transfer functions for the different frequency domains for both collocated and noncollocated transfer functions are summarized in Table G.7. At intermediate frequencies, the motion is governed by the flexible-body modes. At very high frequencies only the core body moves.

G.11.5 Lead compensation of the minimum-phase system

Our goal is to provide a well-damped response to an input command. Therefore it is required to add damping to the double pole at the origin and to add some damping to

```
I1    = 1;
I2    = 0.2;
zeta  = 0.005;
omega = 1;
k     = I2*omega^2;
gamma = (I1 + I2)/I1;
c     = 2*zeta*omega*I2;
num   = [c/I2 k/I2];
den   = [I1 I1*c*gamma/I2 I1*k*gamma/I2 0 0];
FResp(num,den)
```



Example G.15: Bode plot for the noncollocated sensor and actuator transfer function.

Table G.7 Transfer-function summary.

Frequency	Collocated	Noncollocated
Low	$\frac{1}{(I_1+I_2)s^2}$	$\frac{1}{(I_1+I_2)s^2}$
Intermediate	$\frac{s^2+\frac{c}{I_2}s+\frac{k}{I_2}}{I_1 s^2(s^2+c\frac{\gamma}{I_2}s+k\frac{\gamma}{I_2})}$	$\frac{1}{I_1 s^2(s^2+c\frac{\gamma}{I_2}s+k\frac{\gamma}{I_2})}$
High	$\frac{1}{I_1 s^2}$	$\frac{1}{I_1 s^2}$

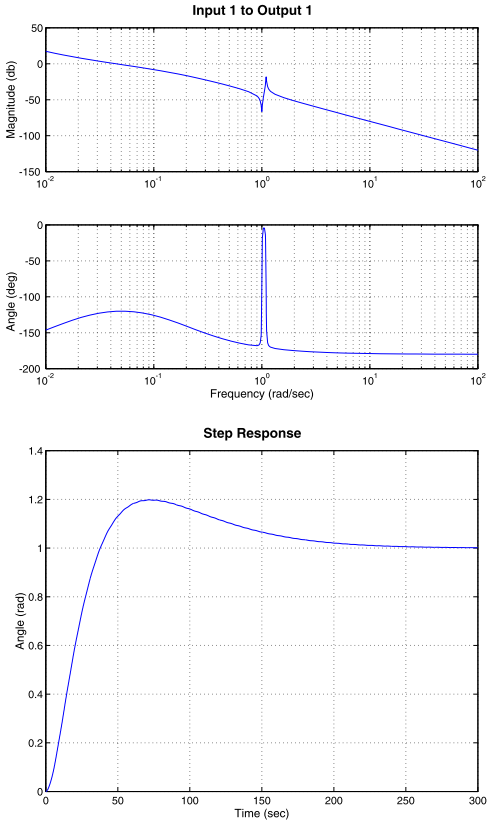
the flexible mode. We have three degrees-of-freedom in the lead compensator. We can adjust the gain, we can choose the maximum amount of phase lead it can add, and we can select the frequency where the phase lead is at its maximum.

If we were not concerned with damping the flex mode, we would put our maximum phase shift at the gain crossover. However, if we want to dampen the flex modes, we want to add some phase lead where they have their gain crossovers. There are three gain crossovers, one at a frequency lower than the resonance and two associated with the resonance pole.

The first step is just to introduce our flex mode into the system for which we have designed the controller above. The result is that the damping at the crossover changes little but the flex-mode damping is still very light. We have not done anything to suppress the flex mode. If we push the lead up to 0.5 rad per second we can add phase lead to the flex mode.

This simple compensator is not the best compensator that could be designed for this system. However, it serves to demonstrate what can be achieved with a very simple design approach. Even with this compensator, the four transient responses do not

```
I1     = 1;
I2     = 0.2;
zeta   = 0.005;
omega  = 1;
k      = I2*omega^2;
gamma  = (I1 + I2)/I1;
c      = 2*zeta*omega*I2;
num    = [1 c/I2 k/I2];
den    = [I1 I1*c*gamma/I2 I1*k*gamma/I2 0 0];
[aP, bP, cP, dP] = ND2SS( num, den );
% Lead
[aL, bL, cL, dL] = LeadLag( 0.05, 60, -63 );
[a,b,c,d]        = Series(aL,bL,cL,dL,aP,bP,cP,dP
    );
FResp(a,b,c,d);

s    = eig(a-b*c)
zeta = S2Damp(s)
g    = statespace(a-b*c,b,c,d);
dT   = 0.1;
nSim = 3000;
y    = Step( g, 1, dT, nSim );
t    = (0:(nSim-1))*dT;
Plot2D( t, y, 'Time (sec)', 'Angle (rad)', 'Step
      Response' );
```



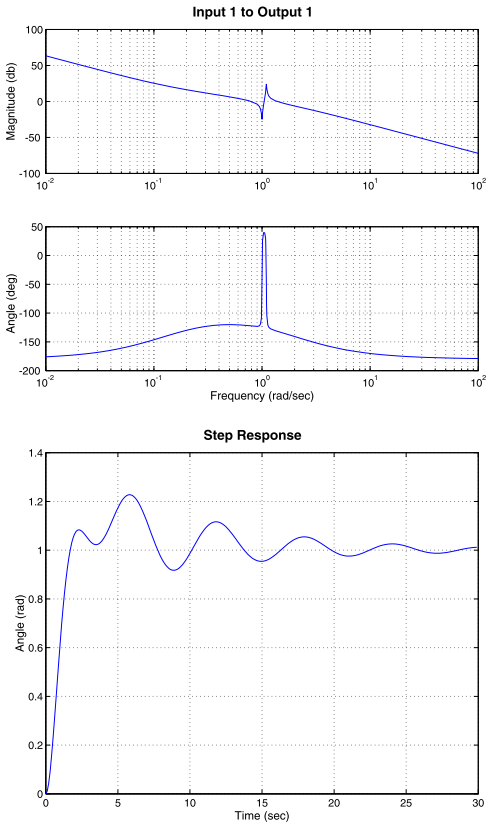
Example G.16: Lead compensator with flex node added.

necessarily show the optimal response. One would not leave the maximum phase lead frequency fixed as the gain and crossover were increased. Rather, one would push the lead frequency up. This would eliminate the oscillatory transient response, but would not increase the damping on the flex mode. To push the bandwidth up and increase the damping on the flex mode, a different compensation approach would be required.

G.11.6 Noncollocated sensor and actuator

Example G.18 shows the Bode plot for a noncollocated sensor and actuator. Example G.18 shows the frequency response over a wider range to show the effect of the high-frequency zero. As expected, it reduces the phase lag to -270° at high frequencies. Clearly, at frequencies below the resonance, the control problem is the same as for the collocated sensor and actuator. To show this, use a phase-lead controller with a crossover at 0.05 rad/s. The Bode plot is shown in Example G.19. -62 dB was chosen for the forward gain. The lead introduced by the controller stabilizes both the control crossover at

```
I1     = 1;
I2     = 0.2;
zeta   = 0.005;
omega  = 1;
k      = I2*omega^2;
gamma  = (I1 + I2)/I1;
c      = 2*zeta*omega*I2;
num    = [1 c/I2 k/I2];
den    = [I1 I1*c*gamma/I2 I1*k*gamma/I2 0 0];
[aP, bP, cP, dP] = ND2SS( num, den );
% Lead
[aL, bL, cL, dL] = LeadLag( 0.5, 60, -15 );
[a,b,c,d]        = Series(aL,bL,cL,dL,aP,bP,cP,dP
    );
FResp(a,b,c,d);

s    = eig(a-b*c)
zeta = S2Damp(s)
g    = statespace(a-b*c,b,c,d);
dT   = 0.1;
nSim = 300;
y    = Step( g, 1, dT, nSim );
t    = (0:(nSim-1))*dT;
Plot2D( t, y, 'Time (sec)', 'Angle (rad)', 'Step
    Response' );
```



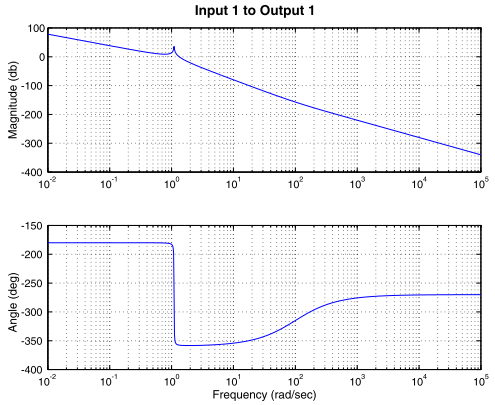
Example G.17: Lead compensator providing flex damping.

0.05 rad/s and the first resonance crossover. The second crossover is stable because the gain is less than 0 dB. This controller has about 16 dB of margin. Example G.20 shows the root-locus plot. The system has two zeros at infinity, a high-frequency plant zero and the controller zero. The two rigid-body poles at the origin go around the controller zero, coalesce on the real axis then one goes to the plant zero and the other goes to the controller zero. The two flex poles go to the zeros at infinity using the right half-plane. They cross into the right half-plane when the gain is about -46 dB.

The controller, even at low gains, does not do anything to improve the damping on the flex mode. At best, it ignores it. Unlike the collocated case, the flex poles do not conveniently approach the open-loop zeros. Instead, they go unstable.

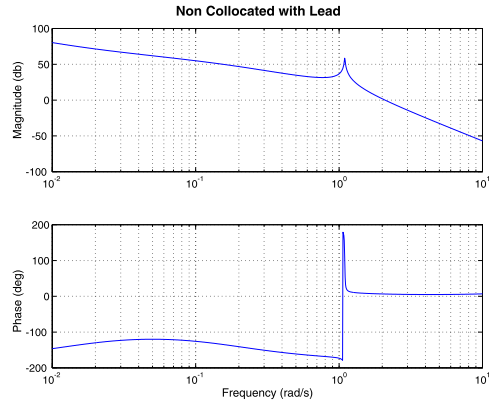
Ignoring the flex mode may be all that is needed to meet specifications. If that is the case the simple controller will do. Suppose, however, that you want your crossover to be above 1 rad/s? The simple lead compensator will not work since it is only supplying a maximum of 60° of phase lead. To compensate this plant, we need a pair of zeros

```
I1     = 1;
I2     = 0.2;
zeta   = 0.005;
omega  = 1;
k      = I2*omega^2;
gamma  = (I1 + I2)/I1;
c      = 2*zeta*omega*I2;
num    = [c/I2  k/I2];
den    = [I1  I1*c*gamma/I2  I1*k*gamma/I2  0  0];
w      = logspace(-2,5,1000);
FResp(num,den,w)
```



Example G.18: Bode plot for the noncollocated sensor and actuator transfer function.

```
I1     = 1;
I2     = 0.2;
zeta   = 0.005;
omega  = 1;
k      = I2*omega^2;
gamma  = (I1 + I2)/I1;
c      = 2*zeta*omega*I2;
num    = [c/I2  k/I2];
den    = [I1  I1*c*gamma/I2  I1*k*gamma/I2  0  0];
[aP, bP, cP, dP] = ND2SS( num, den );
wLead = 0.05;
[aL, bL, cL, dL] = LeadLag( wLead, 60, 0 );
[a,b,c,d] = Series(aL,bL,cL,dL,aP,bP,cP,dP);
w      = logspace(-2,1,1000);
[m,p] = FResp(a,b,c,d,1,w);
yL     = {'Magnitude (db)' 'Phase (deg)'};
Plot2D(w,[20*log10(m);p],'Frequency (rad/s)',yL,'
       Non Collocated with Lead','xlog');
```



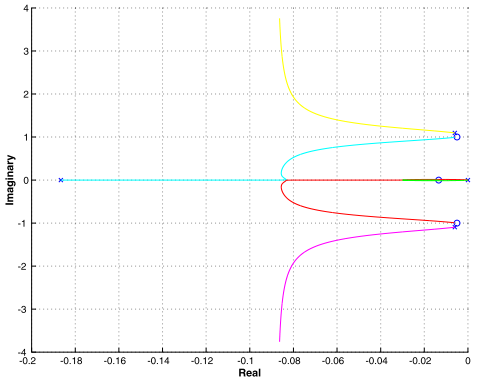
Example G.19: Bode plot for the noncollocated sensor and actuator transfer function with a phase-lead controller and a crossover at 0.05 rad/s.

to attract the flex poles. However, every pair of zeros requires another pair of poles, otherwise, the controller would not be proper. In this plant just adding a pair of zeros (in addition to the lead compensator) would mean that at high frequencies the plant rolls off like 1/s, which is inadequate when unmodeled dynamics are considered. The flex compensator will have the form

$$\frac{s^2 + 2\zeta_1\omega_1 s + \omega_1^2}{s^2 + 2\zeta_2\omega_2 s + \omega_2^2} \tag{G.178}$$

Start by putting a lead compensator with a crossover at 2 rad/s. As expected, the resonance is now unstable since at the phase crossover the gain is more than 1. We need

```
I1     = 1;
I2     = 0.2;
zeta   = 0.005;
omega  = 1;
k      = I2*omega^2;
gamma  = (I1 + I2)/I1;
c      = 2*zeta*omega*I2;
num    = [1 c/I2 k/I2];
den    = [I1 I1*c*gamma/I2 I1*k*gamma/I2 0 0];
[aP, bP, cP, dP] = ND2SS( num, den );

[aL, bL, cL, dL] = LeadLag( 0.05, 60, 0 );
[a,b,c,d] = Series(aL,bL,cL,dL,aP,bP,cP,dP);
g         = statespace(a,b,c,d);
RootLocus( g, logspace(-8,0,10000) );
```



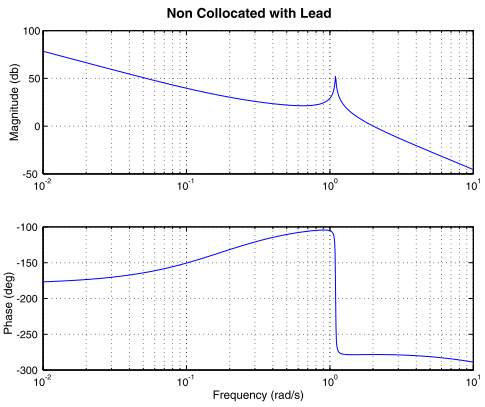
Example G.20: Root-locus plot for the noncollocated sensor and actuator transfer function with a phase-lead controller.

to advance the phase by nearly 180° in the vicinity of the resonance, and the controller crossover.

To get the proper phase advance requires that the numerator of Eq. (G.178) be at a lower frequency than the denominator. The zeros should also be at a lower frequency than the resonance to get the full effect of the zeros before hitting the resonant poles. The appropriate pole and zero frequencies can be achieved with poles or zeros in either the right half-plane or the left half-plane.

Example G.21 shows the crossover at 2 rad/s. The system is unstable due to the phase advance of the zero. Physically this means that we are trying to move the system faster than we can read the response across the flexible connection.

```
I1     = 1;
I2     = 0.2;
zeta   = 0.005;
omega  = 1;
k      = I2*omega^2;
gamma  = (I1 + I2)/I1;
c      = 2*zeta*omega*I2;
num    = [c/I2 k/I2];
den    = [I1 I1*c*gamma/I2 I1*k*gamma/I2 0 0];
[aP, bP, cP, dP] = ND2SS( num, den );
wLead = 2
[aL, bL, cL, dL] = LeadLag( wLead, 80, 0 );
[a,b,c,d] = Series(aL,bL,cL,dL,aP,bP,cP,dP);
w      = logspace(-2,1,1000);
[m,p] = FResp(a,b,c,d,1,1,w);
yL     = {'Magnitude (db)' 'Phase (deg)'};
j      = find(p >= 0 );
p(j)   = p(j) - 360;
Plot2D(w,[20*log10(m);p],'Frequency (rad/s)',yL,'
       Non Collocated with Lead','xlog');
```



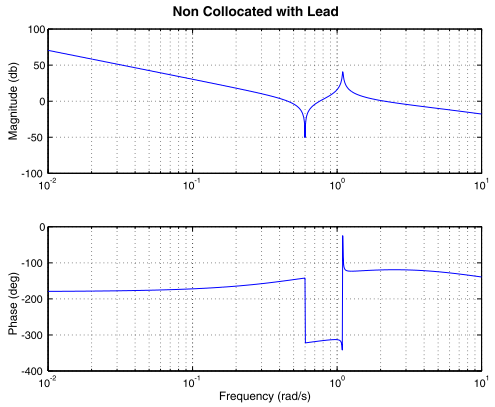
Example G.21: Bode plot with a high-frequency phase-lead controller.

The question of where to put the zeros has to do with our knowledge of the system. Try two different flex compensators of the form

$$\frac{s^2 + (1 \pm 0.4)^2}{s^2 + 0.2 \times 30 + 30^2} \tag{G.179}$$

The damping in the denominator is necessary to stabilize the high-frequency pole introduced by the compensator. This pole is necessary for the compensator to be implementable, you cannot just add the zeros by themselves. The controller pole is high frequency and will damp quickly even with a modest damping ratio. Example G.22 shows that the 2nd-order transfer function provides enough phase shift to result in a stable compensator.

```
I1    = 1;
I2    = 0.2;
zeta  = 0.005;
omega = 1;
k     = I2*omega^2;
gamma = (I1 + I2)/I1;
c     = 2*zeta*omega*I2;
num   = [c/I2 k/I2];
den   = [I1 I1*c*gamma/I2 I1*k*gamma/I2 0 0];
[aP, bP, cP, dP] = ND2SS( num, den );
wLead = 2
[aL, bL, cL, dL] = LeadLag( wLead, 80, 0 );
[a,b,c,d] = Series(aL,bL,cL,dL,aP,bP,cP,dP);
w     = logspace(-2,1,1000);
[m,p] = FResp(a,b,c,d,1,1,w);
yL    = {'Magnitude (db)' 'Phase (deg)'};
j     = find(p >= 0 );
p(j)  = p(j) - 360;
Plot2D(w,[20*log10(m);p],'Frequency (rad/s)',yL,'
       Non Collocated with Lead','xlog');
```



Example G.22: Bode plot with a flex compensator zero at 0.6 rad/s.

G.12. Model-following control

Model following is an approach in which the system is driven to follow a desired model. This approach is demonstrated using a single integrator as an example. The plant is

$$G(s) = \frac{1}{s} \tag{G.180}$$

It is desired that the plant appears as

$$M(s) = \frac{1}{\tau s + 1} \tag{G.181}$$

The following block diagram in Fig. G.10 shows a model-following controller. u is the command and is fed both to the plant and the model. The closed-loop transfer function is

$$\frac{y}{u} = \frac{\tau s + 1 + K}{(\tau s + 1)(s + K)} \tag{G.182}$$

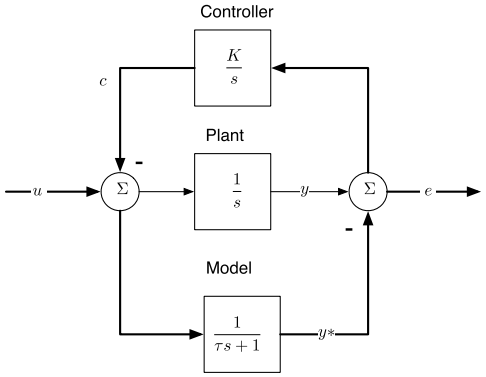


Figure G.10 Model-following controller.

We see that as $K \to \infty$, the transfer function approaches:

$$\frac{y}{u} = \frac{1}{\tau s + 1} \tag{G.183}$$

For finite values of K the damping ratio and undamped natural frequency are

$$\omega_c = \sqrt{\frac{K}{\tau}} \tag{G.184}$$

$$\zeta = \frac{K\tau + 1}{2\sqrt{K\tau}} \tag{G.185}$$

It follows that the system is overdamped ($\zeta \geq 1$) when $K \geq 1/\tau$. The same response could be obtained using the simple feedback controller shown in Fig. G.11. The result-

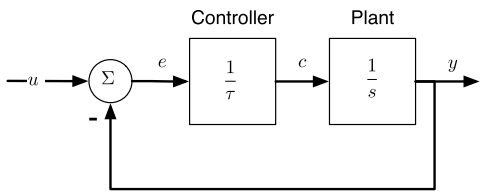


Figure G.11 Feedback controller.

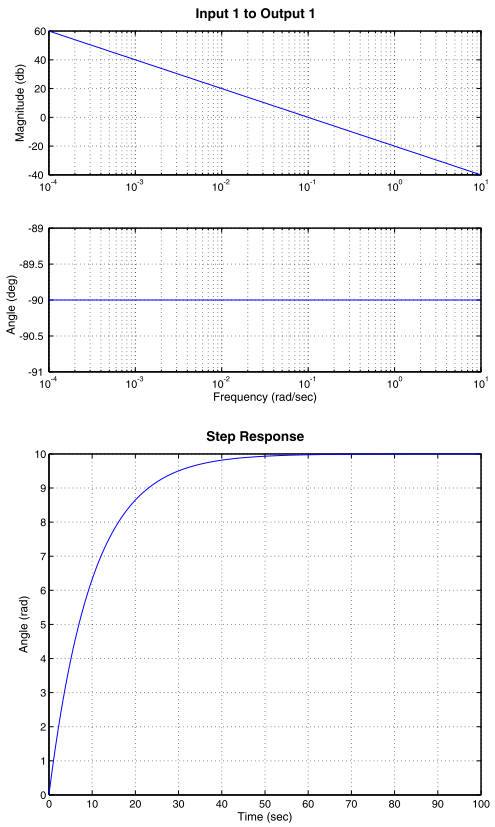
ing transfer function is

$$\frac{y}{u} = \frac{1}{\tau s + 1} \tag{G.186}$$

In this case, the desired response is achieved exactly with simpler constant-gain feedback. Therefore you would not use a model-following control in such a simple system. It is most useful in more complex situations and has been employed successfully on operational systems.

The loop-gain Bode plot and step response of the feedback controlled system are shown in Example G.23. The phase is constant at −90 deg, and the gain has a constant slope of -20 db/decade. This gives an infinite gain margin, since the phase never crosses 180 deg, and a phase margin of 90 deg.

```
% Plant
numP    = 1;
denP    = [1 0];
[aP,bP,cP,dP] = ND2SS( numP, denP );

% Controller
tau   = 10;
K     = 1/tau;

% Forward Loop Gain
[a,b,c,d] = Series(0,0,0,K,aP,bP,cP,dP);

FResp(a,b,c,d);

% Closed Loop System
[aC,bC,cC,dC] = CLoopS( aP,bP,cP, K );

g    = statespace(aC,bC,cC,dC);
dT   = 0.1;
nSim = 1000;
y    = Step( g, 1, dT, nSim );
t    = (0:(nSim-1))*dT;
Plot2D( t, y, 'Time (sec)', 'Angle (rad)', 'Step
      Response' );
```

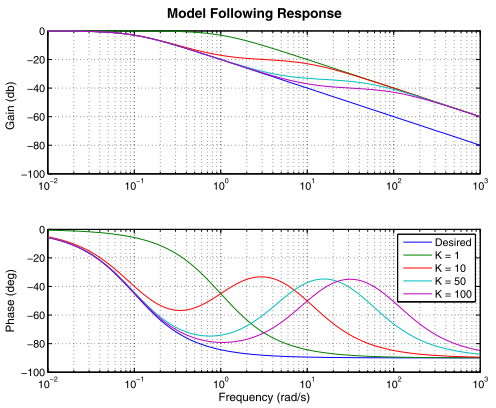


Example G.23: Model-following control.

The Bode plot in Example G.24 shows the effect of the gain K on the control response. Here, we compare the desired response to the response of the closed-loop system associated with the model-following control of Fig. G.10. The closed-loop transfer function of the controlled system is given in (G.182). The controlled system matches the desired response at low frequencies, but a shift in magnitude and phase occurs at

high frequencies. For high gain, the model-matching controller approaches the model, pushing the location of this magnitude and phase shift to a higher frequency.

```
tau = 10;
K   = [1 10 50 100];
w   = logspace(-2,3,100);
m   = zeros(length(K)+1,100);
p   = zeros(length(K)+1,100);
[a,b,c,d] = ND2SS(1,[tau 1]);    % desired
        response
[m(1,:),p(1,:)] = FResp(a,b,c,d,1,1,w);
for i=1:length(K)
   num = [tau K(i)];
   den = [tau tau*K(i)+1 K(i)];
   [a,b,c,d] = ND2SS( num,den );
   [m(i+1,:),p(i+1,:)]=FResp(a,b,c,d,1,1,w);
end



yL = {'Gain (db)', 'Phase (deg)'};
Plot2D( w, [20*log10(m);p], 'Frequency (rad/s)'
        ,...
            yL, 'Model Following Response', 'xlog
                ', {'[1:5]' '[6:10]'})
legend('Desired', 'K=1', 'K=10', 'K=50', 'K
        =100')
```



Example G.24: Model-following control response with increasing gain.

G.13. Double-integrator control

G.13.1 Introduction

The simplest model for many vehicles is a double integrator. This is the equation

$$m\ddot{x} = F \tag{G.187}$$

for linear motion where x equals position, m equals mass, and F equals force,

$$I\ddot{\theta} = T \tag{G.188}$$

for rotational motion where θ equals angle, I inertia, and T equals torque. We can write both equations as

$$\ddot{w} = u \tag{G.189}$$

where u is acceleration and w is the state. A step acceleration produces the response

$$w = \frac{1}{2}ut^2 + \dot{w}(0)t + w(0) \tag{G.190}$$

which shows that w grows linearly with the initial time derivative and with the square of time multiplied by the acceleration. If u is a control this response is fine. If u is a disturbance we do not want w to change and so must apply control. Interestingly,

(G.190) is the basis for bang-bang control. That is you command u with one sign then when w reaches the half-way point, switch the sign of w. This will be discussed later.

In this section we will first discuss linear control techniques and then discuss the mathematical basis for bang-bang control. The system is illustrated in Fig. G.12.

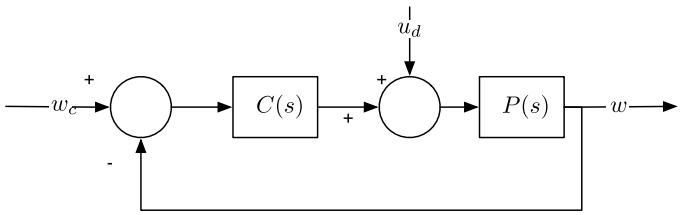


Figure G.12 Block diagram of the closed-loop control system.

The response of w to disturbances and control inputs is

$$w = Pu_d + PC(w_c - w) \tag{G.191}$$

or

$$\frac{w}{u_d} = \frac{P}{1 + PC} \tag{G.192}$$

$$\frac{w}{w_c} = \frac{PC}{1 + PC} \tag{G.193}$$

In both cases, the denominator is $1 + PC$ but the numerators for the command input, w_c, and the disturbance input, u_d are different. For the former, if PC is much greater than 1 the output exactly follows the input. This can be done by letting $C = K/P$ and making K large. This is known as inverting the plant but requires ∞ energy. PC is the open-loop response, that is the series connection of controller and plant.

G.13.2 Linear control

The first thing to try is proportional control

$$\ddot{w} + \sigma^2 w = u_d \tag{G.194}$$

where ω will be the oscillation frequency of the system. It is instrumental to look at the double-integrator and oscillator-frequency responses. The two transfer functions are

$$\frac{w(s)}{u(s)} = \frac{1}{s^2} \tag{G.195}$$

$$\frac{w(s)}{u(s)} = \frac{1}{s^2 + \sigma^2} \tag{G.196}$$

The first has a singularity at $s = j_0$. The second has a singularity at $s = j_\sigma$. At high frequencies, they have the same response $\frac{1}{s^2}$. At j_0 the second has the response $\frac{1}{\sigma^2}$, thus the wider the bandwidth of the controller (determined by σ) the smaller the response.

The infinite responses in these two cases mean that the system will absorb all of the input at those frequencies and as $t \to \infty$ the response will go to ∞. From a practical point of view, it means the system will oscillate at σ, which is usually undesirable.

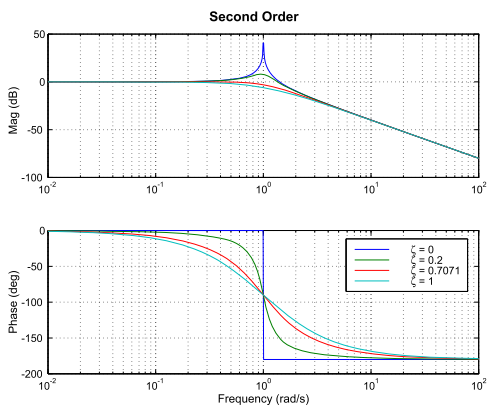
Since the oscillatory response is undesirable in most cases the next step is to add damping. This is done by adding a control term proportional to the derivative of w

$$\ddot{w} + 2\zeta\sigma w + \sigma^2 w = u_d \tag{G.197}$$

where ζ is the damping ratio that goes from 0 to ∞. The transfer function is now

$$\frac{w(s)}{u(s)} = \frac{1}{s^2 + 2\zeta\sigma s + \sigma^2} \tag{G.198}$$

```
sigma = 1;
zeta  = [0 0.2 0.7071 1];
w     = logspace(-2,2,1000);
for k = 1:length(zeta)
  [aP, bP, cP, dP] = ND2SS( 1, [1 2*zeta(k)*sigma
        sigma^2] );
  [m(k,:), p(k,:)] = FResp(aP,bP,cP,dP,1,1,w);
end
yL = {'Mag (dB)' 'Phase (deg)'};
j  = find(p > 0);
p(j) = p(j) - 360;
Plot2D(w,[20*log10(m);p],'Frequency (rad/s)',yL,'
      Second Order','xlog', {'[1:4]' '[5:8]'});
legend('\zeta=0','\zeta=0.2', '\zeta=0.7071
      ', '\zeta=1')
```



Example G.25: Second-order response.

When $\zeta = 1$ the resonant peak is gone. The trade for higher ζ is that there is more phase shift at lower frequencies. This means the control will be more sluggish. Note that at high frequencies the response is the same as for a double integrator, that is -180° and the magnitude decreases as $1/\omega^2$. This just means that the control has no effect at frequencies much higher than σ.

Adding damping has not changed the response to a step disturbance, which will still be $\frac{1}{\sigma^2}$. If this offset is unacceptable we must add an integrator to the controller, which will result in what is known as PID (proportional, integral, differential) control. There

are many ways to formulate the control but one is

$$\frac{u_c(s)}{w(s)} = K_p\left(1 + \tau_d s + \frac{1}{\tau_i s}\right) \tag{G.199}$$

where τ_d is the rate time constant that is how long the system will take to damp and τ_i is how fast the system will integrate out a steady disturbance.

The open-loop transfer function is

$$\frac{w(s)}{u_d(s)} = \frac{K_p}{s^2}\left(1 + \tau_d s + \frac{1}{\tau_i s}\right) \tag{G.200}$$

The closed-loop transfer function is

$$\frac{w(s)}{u_d(s)} = \frac{s}{s^3 + K_p\tau_d s^2 + K_p s + K_p/\tau_i} \tag{G.201}$$

The desired closed-loop transfer function is

$$\frac{w(s)}{u_d(s)} = \frac{s}{(s+\gamma)(s^2 + 2\zeta\sigma s + \sigma^2)} \tag{G.202}$$

or

$$\frac{w(s)}{u_d(s)} = \frac{s}{s^3 + (\gamma + 2\zeta\sigma)s^2 + \sigma(\sigma + 2\zeta\gamma)s + \gamma\sigma^2} \tag{G.203}$$

The parameters are

$$K_p = \sigma(\sigma + 2\zeta\gamma) \tag{G.204}$$

$$\tau_i = \frac{\sigma + 2\zeta\gamma}{\gamma\sigma} \tag{G.205}$$

$$\tau_d = \frac{\gamma + 2\zeta\sigma}{\sigma(\sigma + 2\zeta\gamma)} \tag{G.206}$$

It is interesting to evaluate the effect of the integrator. This is done in Example G.26. The smaller γ, which is the inverse of the integrator time constant, the longer it takes to integrate out the disturbance.

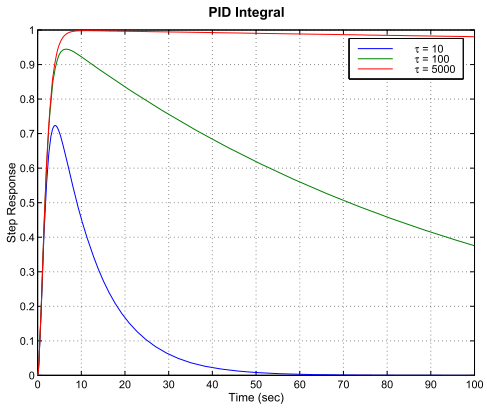
The final form of the PID is shown below,

$$T = K_p + K_R\frac{\omega_R s}{s + \omega_R} + \frac{K_I}{s} \tag{G.207}$$

The derivative term is multiplied by a first-order filter to prevent differentiation at high frequencies. Rearranging gives

$$\frac{T}{\theta} = \frac{(K_P + K_R\omega_R)s^2 + (K_p\omega_R + K_I)s + (K_P + K_i\omega_R)}{s(s + \omega_R)} \tag{G.208}$$

```
gamma    = 1./[10  100  1000];
dT       = 0.1;
nSim     = 1000;
y        = zeros(3,nSim);
for k = 1:length(gamma)
  [a,b,c,d] = ND2SS( [1  0], [1 (gamma(k) + 2) (1
        + 2*gamma(k)) gamma(k)]);
  g          = statespace(a,b,c,d);
  y(k,:)     = Step( g, 1, dT, nSim );
end
t = (0:(nSim-1))*dT;
Plot2D(t, y, 'Time (sec)', 'Step Response', 'PID
      Integral');
legend('\gamma = 0.1','\gamma = 0.01', '\gamma =
      0.001')
```



Example G.26: Effect of the integrator on a second-order step response.

If ω_R is set to ∞ this becomes

$$\frac{T}{\theta} = \frac{K_R s^2 + K_P s + K_I}{s} \tag{G.209}$$

This transfer function is not proper, i.e., it goes to ∞ as s goes to 0. The latter cannot be implemented in state-space form for this reason. In practice, this is not a problem if noise filtering is included in the loop. The built-in rate filter is often sufficient. On occasion, it may be necessary to add a notch filter to notch out a flex or slosh mode. A low-pass filter may also be added if the sensor is noisy.

G.13.3 Phase-plane controller

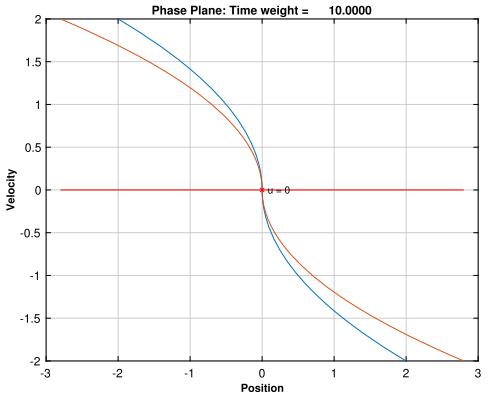
Another way to control a double integrator is a phase-plane controller. This is just the embodiment of the equation

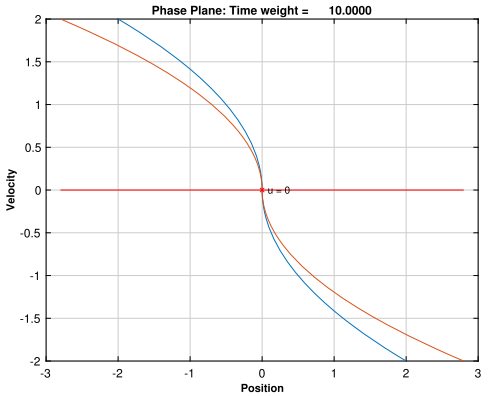
$$x = \frac{1}{2}ut^2 + v(0)t + x(0) \tag{G.210}$$

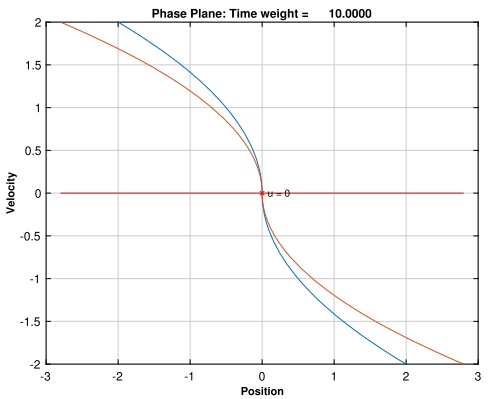
where $\dot{x} = v$ when u is limited to be $\pm u_{max}$. There will be a locus of points $[x, v]$ for which a constant u will drive x and v to zero. This is a switching curve. If we are only solving the time-optimal problem there will be one switching curve. If we solve the fuel–time optimal problem, there will be two switching curves and the trajectory will coast between the curves.

Example G.27 shows a fuel–time optimal pair of switching curves and then compares a PID controller with a phase-plane controller.

```
PhasePlane( [], [],10);
dT              = 0.01;
[a,b,c,d]       = PIDMIMO( 1, 0.7071, 0.1, 100.0,
       5.0, dT, 'Z' );
aP              = [0 1;0 0]; bP = [0;1]; cP = [1
       0]; dP = 0;
[aP,bP]         = C2DZOH( aP, bP, dT );
xPID            = [10;0];
nSim            = 8000;
xPID            = [xPID zeros(2,nSim)];
xPP             = xPID;
xC              = zeros(2,1);
uPID            = zeros(1,nSim+1);
uPP             = zeros(1,nSim+1);

for k = 1:nSim

  % PID Controller
   u  = -c*xC - d*xPID(1,k);
   xC =  a*xC + b*xPID(1,k);
   xPID(:,k+1) = aP*xPID(:,k) + bP*u;
   uPID(:,k+1) = u;

  % Phase Plane
   u = PhasePlane(xPP(2,k),xPP(1,k),0,0.01,0.01);
   xPP(:,k+1) = aP*xPP(:,k) + bP*u;
   uPP(:,k+1) = u;
end

t = (0:nSim)*dT;
Plot2D( [xPID(1,:);xPP(1,:)], [xPID(2,:);xPP(2,:)
       ], 'x', 'v', 'PID' )
legend('PID', 'Phase Plane')
Plot2D( t, [xPID;xPP;uPID;uPP], 'Time (sec)', {'x
       ' 'v' 'uPID' 'uPP'},...
        'Red is the Phase Plane','lin', {'[1 3]'
              '[2 4]' '[5  ]' '[6  ]'} );
```





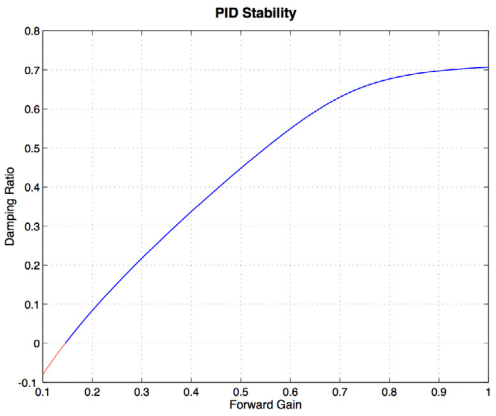


Example G.27: Phase-plane controller.

G.13.4 Control limiting

A PID controller is only conditionally stable. If the forward gain decreases due to actuator saturation, the controller can be unstable. This is sometimes known as integrator windup. This is seen in Example G.28. For a forward gain of less than 0.14 the system is unstable. The unstable region is shown in red.

```
inr = 10000;

[aC, bC, cC, dC] = PIDMIMO( inr, 0.7071, 0.04,
      50, 1.0 );
aP = [0 1;0 0];
bP = [0; 1/inr];
cP = [1 0];
dP = 0;
[a,b,c,d] = Series(aP,bP,cP,dP,aC,bC,cC,dC);

kF = linspace(0.1,1);

for k = 1:length(kF)
  s = eig(a-kF(k)*b*c);
  z = S2Damp(s);
  zeta(k) = z(3);
end
j = find(zeta <= 0);
k = j(end):length(zeta);
Plot2D(kF(k),zeta(k),'Forward Gain','Damping
      Ratio','PID Stability')
hold on
plot(kF(j),zeta(j),'r')
```



Example G.28: PID damping ratio.

Since a PD controller is unconditionally stable the simplest way to deal with actuator saturation is to limit the integral. This can be done for any state-space system. If u is the actuator output write

$$y_k = \text{sat}(cx_k + du_k) \tag{G.211}$$

$$x_{k+1} = (a - lc)x_k + (b - ld)u_k + ly_k \tag{G.212}$$

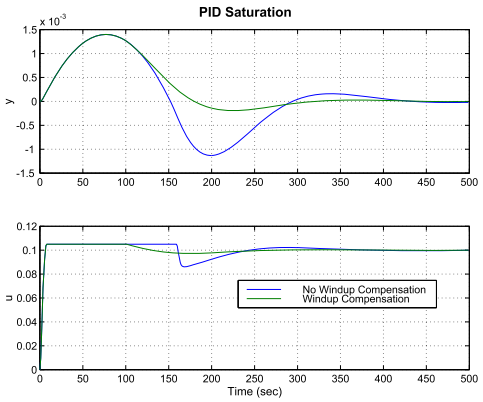
If y is not saturated these equations reduce to

$$y_k = x_k + du_k \tag{G.213}$$

$$x_{k+1} = ax_k + bu_k \tag{G.214}$$

The gain matrix l is chosen so that the eigenvalues of $a - lc$ are all zero, that is it is a deadbeat controller, see Example G.29. Note that the performance of the compensator with windup compensation is superior to that of the compensator without the windup compensation. The uncompensated system effectively looks more lightly damped than that with the windup compensation.

```
dT  = 1.0; inr = 10000;

[aC, bC, cP, dP] = PIDMIMO( inr(1,1), ...
                            0.7071,...
                            0.04, 50, ...
                            1.0, dT, 'z' );
[aP,bP] = C2DZOH( [0 1;0 0], [0; 1/inr(1,1)], dT
      );

nSim    = 500;
uStep   = 0.1; uSat = 0.105;
uPlot   = zeros(2,nSim);
yPlot   = zeros(2,nSim);

for j = 1:2
  x = zeros(2,1);
  y = 0; xPID = [0;0];

  if( j == 2 )
    [aC,bC,lC] = Windup( aC, bC, cC, dC );
  else
    lC = [0;0];
  end

  for k = 1:nSim
    % Controller
    %------------
    u = cC*xPID + dC*y;
    if( abs(u) > uSat ) u = sign(u)*uSat; end

    xPID = aC*xPID + bC*y + lC*u;

    yPlot(j,k) = y; uPlot(j,k) = u;

    % The plant
    %----------
    x = aP*x + bP*(uStep-u);
    y = x(1);
  end
end

Plot2D((0:(nSim-1))*dT,[yPlot;uPlot],'Time (sec)'
      ,['y';'u'],...
       'PID Saturation','lin',['1:2';'3:4']);
legend('No Windup Compensation','Windup
      Compensation')
PrintFig(1,1,1,'PIDWindup')
```



Example G.29: Windup compensation.

G.13.5 Cross-axis coupling

It is often the case that multiple axes must be controlled. Given a system of the form

$$M\ddot{x} = F \tag{G.215}$$

where M is not diagonal we define F as

$$F = Ma \tag{G.216}$$

where a is the vector of accelerations. Then, our control problem is the decoupled problem,

$$\ddot{x} = a \tag{G.217}$$

so we can solve three decoupled PID problems.

G.14. Lyapunov control

G.14.1 Background

Systems that are time invariant are known as autonomous systems. Those that are time varying are called nonautonomous systems,

$$\dot{x} = f(x) \tag{G.218}$$

is an autonomous system

$$\dot{x} = f(x, t) \tag{G.219}$$

is nonautonomous. Adding a controller

$$\dot{x} = f(x, u) \tag{G.220}$$

can make an autonomous system nonautonomous if $u(t)$. For example, a tracking controller is nonautonomous since the set point varies with time. Systems with external disturbances are nonautonomous.

G.14.2 Theory

Lyapunov's control theory consists of two methods: Lyapunov's first method that involves linearization about an operating point and Lyapunov's direct method that applies to nonlinear systems. Lyapunov's direct method is based on the idea that if a system dissipates energy it will eventually reach equilibrium.

Lyapunov's direct method has two steps. The first is to find a scalar function of the states of the system, known as the Lyapunov function, and then to evaluate its derivative along the trajectory of the system. If the Lyapunov function is always decreasing the system will eventually reach equilibrium.

If there exists a Lyapunov function $V(x)$ such that

1. $V(x)$ is positive-definite;
2. $\dot{V}(x)$ is negative-definite;
3. $V(x) \to \infty$ as $||x|| \to \infty$;

then the equilibrium point is uniformly asymptotically stable.

For a linear system

$$\dot{x} = Ax + Bu \tag{G.221}$$

if we define the cost

$$C = \frac{1}{2}\left(x^T Q x + u^T R u\right) \tag{G.222}$$

the solution, P, to the matrix Riccati equation

$$A^T P + PA - PBR^{-1}B^T P - Q = 0 \tag{G.223}$$

can be used in the Lyapunov function

$$V = x^T P x \tag{G.224}$$

and the stabilizing controller is

$$u = -R^{-1} B^T P x \tag{G.225}$$

G.14.3 Nonlinear rate damper

The dynamical equations for a spacecraft are

$$I\dot{\omega} + \omega^{\times} I\omega = T \tag{G.226}$$

Let the control be

$$T_c = -K\omega \tag{G.227}$$

The dynamical equations are now

$$I\dot{\omega} + \omega^{\times} I\omega + K\omega = T_d \tag{G.228}$$

where T_d is the disturbance torque. The kinetic energy is our Lyapunov function and is

$$V = \frac{1}{2}\omega^T I\omega \tag{G.229}$$

The time derivative is

$$\dot{V} = \omega^T I\dot{\omega} \tag{G.230}$$

Substitute in the dynamical equations

$$\dot{V} = \omega^T \left(T_d - (\omega^{\times} I + K)\omega\right) \tag{G.231}$$

Looking at just the damping case in which $T_d = 0$,

$$\dot{V} = -\omega^T \left(\omega^{\times} I + K\right)\omega \tag{G.232}$$

This system is autonomous if I is time invariant. $\dot{V}$ is negative-definite if $-\dot{V}$ is positive-definite, which requires that for all $x \neq 0$ $\dot{V} > 0$. If we linearize this we see that any positive diagonal gain matrix makes this true.

$$K_x\omega_x^2 + K_y\omega_y^2 + K_z\omega_z^2 > 0 \tag{G.233}$$

for all ω.

We can expand this for a diagonal inertia matrix and diagonal gain matrix without a loss of generality. Expanding, we find

$$\omega^T\left(\omega^{\times} I + K\right)\omega = \omega_x\omega_y\omega_z\left[I_z - I_y + I_x - I_z + I_y - I_x\right] + K_x\omega_x^2 + K_y\omega_y^2 + K_z\omega_z^2 \tag{G.234}$$

which equals

$$K_x\omega_x^2 + K_y\omega_y^2 + K_z\omega_z^2 \tag{G.235}$$

G.15. First- and second-order systems

A damped first-order system is

$$\dot{x} + ax = au \tag{G.236}$$

where a is the damping constant, x is the state, and u is the input. When the initial transient is done, $\dot{x} = 0$, the steady-state response is

$$x = u \tag{G.237}$$

A damped second-order system is

$$\ddot{x} + 2\zeta\omega\dot{x} + \omega^2 x = \omega^2 u \tag{G.238}$$

where ζ is the damping ratio, x is the state, ω is the undamped natural frequency, and u is the input. When the initial transient is done, $\dot{x} = 0$ and $\ddot{x} = 0$, the steady-state response is

$$x = u \tag{G.239}$$

G.16. Inner and outer loops

Assume you have a dynamical system composed of two equations for the pitch loop of a momentum-bias spacecraft.

$$T = I\ddot{\theta} + J\dot{\Omega} \tag{G.240}$$

$$T_w = J\dot{\Omega} \tag{G.241}$$

where I is the pitch inertia, J is the momentum wheel inertia, T is an external disturbance torque, θ is the angle that you want to control, Ω is your momentum wheel speed, and T_w is the torque on the wheel produced by an electric motor connecting the

wheel to the spacecraft. You have a measurement of Ω and θ but not T_w. How do we connect the two? Write

$$J\dot{\Omega} = I\left(2\zeta\sigma\dot{\theta} + \sigma^2\theta\right) \tag{G.242}$$

where ζ is the damping ratio and σ is the undamped natural frequency. Our first dynamical equation is now

$$T = I\ddot{\theta} + I\left(2\zeta\sigma\dot{\theta} + \sigma^2\theta\right) \tag{G.243}$$

which is a damped second-order system. Divide by I the clarify to equation

$$\frac{T}{I} = \ddot{\theta} + \zeta\sigma\dot{\theta} + \sigma^2\theta \tag{G.244}$$

Our inner-loop commands wheel speed,

$$T_w = Jk\left(\Omega_c - \Omega\right) \tag{G.245}$$

where k is the gain and Ω_c is the commanded wheel speed. The inner-loop dynamical equation is now

$$k\Omega_c = \dot{\Omega} + k\Omega \tag{G.246}$$

which is a damped first-order system. If k is high enough, that is the inner loop is very fast, the wheel speed will equal the commanded speed.

$$\Omega = \Omega_c \tag{G.247}$$

This is the "steady-state" response. Ω is determined by integrating Eq. (G.242)

$$\Omega = \left(\frac{I}{J}\right)\int\left(2\zeta\sigma\dot{\theta} + \sigma^2\theta\right) \tag{G.248}$$

or

$$\Omega_c = \left(\frac{I}{J}\right)\sigma\left(2\zeta\dot{\theta} + \sigma\int\theta\right) \tag{G.249}$$

Our outer loop will control θ. Our inner loop will control Ω, which is > 0 for all ω. Therefore the damping system will work for the nonlinear system.

APPENDIX H

Estimation theory

There are many ways to take measurements from spacecraft sensors and use them to produce and estimate of the spacecraft's orientation. Kalman filters, while not the only approach, are often a good approach to this problem.

There are several types of attitude estimation algorithms that are based on the Kalman filter. These include (in order of complexity),

1. The Kalman filter;
2. The extended Kalman filter;
3. The unscented Kalman filter.

A Kalman filter consists of a measurement model that relates measurements to the spacecraft state, and a dynamical model of the time evolution of the state. The three types of Kalman filters listed above approach the problem in different ways.

The first method estimates spacecraft attitude using (usually approximate) static, linear relationships between the attitude and sensor measurements. The dynamical model is also linear.

The other two methods can utilize the nonlinear measurement and dynamical models directly. They are much more complex and require more flight software for their implementation.

This chapter will start with estimation system and then move on to Kalman filters.

H.1. Estimation theory

Given a physical system that can be modeled in the form

$$\dot{x} = f(x, u, t) \tag{H.1}$$

the goal is to get the best possible estimate of x. Take, for example, a very simple physical system in continuous time

$$\dot{x} = -ax + u \tag{H.2}$$

$$y = x \tag{H.3}$$

with only one parameter, a, and equations that are first order and linear. The input to the system is u and the system state is x. The sensor measurement y is of the system state, i.e., $y = x$.

A simple estimator algorithm consists of numerically integrating the following dynamical equation:

$$\dot{x}_E = ax_E + u_E + L(y_M - y_E) \tag{H.4}$$

In the above equation subscript E emphasizes that we are integrating estimates of the corresponding variables, and y_M is the actual sensor measurement. We note that, as per Eq. (H.3), $y_E = x_E$, i.e., the estimated (or expected) sensor measurement is the same as the estimated state of the system in this example. L is called the gain of the estimator. Effectively, L finds the difference between the estimated measurement and the actual measurement. It produces an input that is added to u_E that pushes the estimate, x_E towards x.

The transfer function between the estimated x and the measurement y_M and estimated input u_E can be computed by taking Laplace transforms of both sides of Eq. (H.4):

$$x_E(s) = \frac{Ly_M}{s + (a - L)} + \frac{u_E}{s + (a - L)} \tag{H.5}$$

Thus we have a first-order, low-pass filter with a DC gain of $L/(a - L)$ from y_M to x_E and a first-order, low-pass filter with a DC gain of $1/(a_L)$ from u_E to x_E. We note that if $L = 0$ the estimator (Eq. (H.4)) just propagates the estimated inputs. On the other hand, if L is large the estimator ignores the inputs and effectively only uses the measurements. The filter is less responsive to high-frequency components of the input measurement signal – the "cut-off frequency" of the filter is $a - L$.

If the bandwidth of u is known, L can be selected so that frequencies beyond those contained in u would be attenuated. Given a knowledge of the measurement noise and the uncertainty in the plant one can optimally pick L using quadratic estimator theory. The optimal gain is found from the equations

$$\dot{p} = -2ap + q - \frac{p^2}{r} \tag{H.6}$$

$$L = \frac{p}{r} \tag{H.7}$$

$$E[v(t)v^T(t')] = rb(t - t') \tag{H.8}$$

$$E[w(t)w^T(t')] = qb(t - t') \tag{H.9}$$

where $E[.]$ is expectation. Given the model

$$\dot{x} = -ax + u + w \tag{H.10}$$

$$y = x + v \tag{H.11}$$

where w and v are white-noise processes, d has units of seconds, q represents uncertainty in the plant model and in the inputs u, and r represents measurement noise.

In steady state (as t approaches ∞)

$$L = \sqrt{a^2 + q/r} - a \tag{H.12}$$

In the limit

$$L = \begin{cases} 0 & r/q \to \infty \\ \sqrt{q/r} & q/r \to \infty \end{cases} \tag{H.13}$$

This result means that for very noisy signals, or perfect knowledge of the plant, L should be set to zero. This can also be implemented as a time-varying filter. The time-varying gain is

$$L(t) = \beta - a + \frac{2\beta}{\left(\frac{p(0)+r(\beta+a)}{p(0)-r(\beta-a)}\right) e^{2\beta t} - 1} \tag{H.14}$$

$$\beta = \sqrt{a^2 + q/r} \tag{H.15}$$

$L(0) = \frac{p(0)}{r}$, which means that the bigger the uncertainty in x initially, the bigger the gain. This gain is independent of the measurements or the state. The steady-state value is independent of $p(0)$. Therefore given a, q, and r, it will always converge to the same value. This will be true about all time-varying filters derived this way including the extended Kalman filters.

The gain is dependent on the ratio between the plant and measurement uncertainty and the plant parameters. If a time-varying filter is used, the gain will always vary in the same way, given the same initial covariance. The white-noise approximation for the plant noise is usually not very good. It is often necessary to augment the state equations with additional equations modeling the plant and disturbance uncertainty. The ultimate form of the estimator is a first-order, low-pass filter, except that an estimate of the disturbance is added. If this is unavailable the estimator is a simple first-order, low-pass filter.

Usually, data for gyros is provided as spectral densities. Thus these equations can be used directly to determine the covariances of the estimates in steady-state. They provide a useful check for the discrete-time Kalman-filter equations.

The matrix Riccati equation is

$$\dot{P} = FP + PF^T - PH^T R^{-1} HP + Q \tag{H.16}$$

which relates the state covariance, P, to the plant model, F, the measurement matrix, H, the measurement noise matrix, H, and the plant noise spectral density matrix, Q.

In this case, F is the state matrix

$$F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{H.17}$$

and H is the measurement equation

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{H.18}$$

Q is the spectral density matrix for the plant and R is the spectral density matrix for the measurement. Expanding the covariance equation with the derivative set to zero gives

$$\begin{bmatrix} p_{12} & p_{22} \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} p_{12} & 0 \\ p_{22} & 0 \end{bmatrix} - \frac{1}{r}\begin{bmatrix} p_{11}^2 & p_{11}p_{12} \\ p_{11}p_{12} & p_{12}^2 \end{bmatrix} + \begin{bmatrix} q_{11} & 0 \\ 0 & q_{22} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{H.19}$$

Writing out the scalar equations gives

$$2p_{12} + q_{11} - \frac{p_{11}^2}{r} = 0 \tag{H.20}$$

$$p_{22} = \frac{p_{11}p_{12}}{r} \tag{H.21}$$

$$p_{12} = \sqrt{rq_{22}} \tag{H.22}$$

Solving for elements of P,

$$p_{11} = \sqrt{r(q_{11} + 2\sqrt{q_{22}})} \tag{H.23}$$

$$p_{22} = \sqrt{\frac{q_{22}(q_{11} + 2\sqrt{q_{22}})}{r}} \tag{H.24}$$

$$p_{12} = \sqrt{rq_{22}} \tag{H.25}$$

If there were no plant noise, i.e., if $q_{11} = q_{22} = 0$ the steady-state covariance matrix would go to zero, as would be expected.

H.1.1 Conversion from continuous to discrete time

The continuous-time data assumes continuous measurements. In practice, the sensors are sampled. Converting the continuous spectral densities to covariances is done by

$$Q_k = \tau Q \tag{H.26}$$

$$R_k = \frac{R}{\tau} \tag{H.27}$$

where t is the sampling time. In this case, the units of Q_k and R_k are rad^2 and (rad/s)^2.

H.2. The Kalman-filter algorithm

Kalman filters are recursive estimators that adjust their gains based on a model for the system. Essentially, that automates the process of finding the passband for a system. The simplest Kalman filter is for a continuous linear dynamical model, known as the plant, of the form

$$\dot{x} = ax + bu \tag{H.28}$$

$$y = cx \tag{H.29}$$

where a and b are transformed to discrete time given the desired time step. The time step needs to be short enough so that the frequency

$$\frac{2\pi}{\Delta t} \tag{H.30}$$

is higher than the dynamics of a. Discretizing can be done in many ways. In the book, we use the zero-order hold (ZOH). This forms the basis of the discrete-time linear Kalman filter,

$$x_{k+1} = \Phi x_k + \Gamma u_k + \eta_x \tag{H.31}$$

$$y_k = Hx_k + \eta_y \tag{H.32}$$

where x is the state, Φ is the state-transition matrix, Γ is the input matrix, y_k is the measurement, and H is the measurement matrix. H and c are the same. Assume that there is a Gaussian white-noise input to the plant and Gaussian white noise on the measurement. The plant white noise is due to unmodeled inputs and modeling errors. Modeling errors are due to several reasons

1. A reduced-order model is used;
2. There is uncertainty in the values for Φ, Γ, and H matrices;
3. The system is nonlinear and the linear matrices are only valid at the operating point;
4. There are poorly known inputs to the system.

The Kalman filter has two parts. One is an update due to the measurement. The second is due to the prediction of change in the state, x, due to the dynamics. We leave out the k indices,

$$K = \frac{pH^T}{HPH^T + R} \tag{H.33}$$

$$x_e = x_e + K(y - Hx_e) \tag{H.34}$$

$$P = (E - KH)P \tag{H.35}$$

where E is the identity matrix, Q is the covariance matrix for the model uncertainty, P is the state covariance matrix, and R is the measurement covariance matrix. x_e is the

estimated state and y is the measurement.

$$x_e = \Phi x_e + \Gamma u \tag{H.36}$$

$$P = \Phi P \Phi^T + \Gamma Q \Gamma^T \tag{H.37}$$

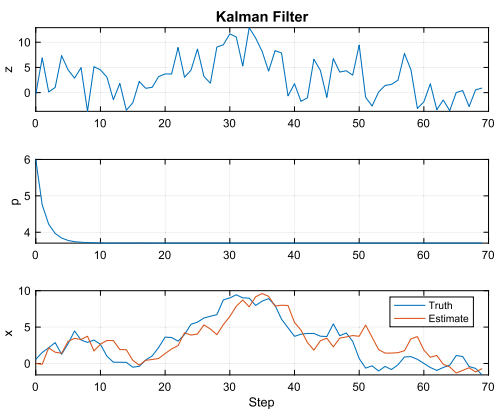
The code in SimpleKalmanFilter is for the system

$$x_{k+1} = x_k + u_k \tag{H.38}$$

$$y_k = x_k \tag{H.39}$$

where x is a scalar. This a single integrator. The results, which are dependent on the covariances, are shown in Example H.1. The estimate tracks the state reasonably well. The covariance reaches a predetermined value based on the ratio of Q and R. The covariance is like clockwork, it is independent of the measurements.

```
n     = 70; % Number of steps
p0    = 6; % Initial covariance
r     = 10; % Measurement noise
q     = 1; % Plant noise
h     = 1; % Measurement matrix
phi   = 1; % State transition matrix
gamma = 1; % Input matrix
x     = 1; % Initial state
xE    = 0; % Estimated state
u     = 0; % Input
p     = p0;

xP    = zeros(4,n);
for j = 1:n

        % Truth model
  z = h*x + sqrt(r)*randn; % Measurement
  x = phi*x + gamma*u + sqrt(q)*randn;

  % Store variables to plot
  xP(:,j) = [z;p;x;xE];

  % Kalman Filter

  % Update
  k  = p*h'/(h*p*h' + r);
  xE = xE + k*(z - h*xE);
  p  = (1 - k*h)*p; % Measurement update of
       covariance

  % Prediction
  xE = phi*xE + gamma*u;
  p  = phi*p*phi' + gamma*q*gamma';

end

yL = {'z' 'p' 'x'};
leg = {{},{},{'Truth','Estimate'}};

Plot2D(0:n-1,xP,'Step',yL,'Kalman Filter',...
       'lin',{'1' '2' '[3 4]'},[],[],[],[], ...
       leg);
```



Example H.1: Single-integrator Kalman-filter results.

H.3. Bayesian derivation

Kalman filters were originally derived using least-squares. Kalman filters can also be derived from the Bayes theorem. What is Bayes theorem? Bayes theorem is

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum P(B|A_i)} \tag{H.40}$$

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)} \tag{H.41}$$

which is just the probability of A_i given B. P means "probability". The vertical bar | means "given". This assumes that the probability of B is not zero, that is $P(B) \neq 0$. In the Bayesian interpretation, the theorem introduces the effect of evidence on belief. This provides a rigorous framework for incorporating any data for which there is a degree of uncertainty. Put simply, given all the evidence (or data) to date, Bayes theorem allows you to determine how new evidence affects the belief. In the case of state estimation, this is the belief in the accuracy of the state estimate.

Fig. H.1 shows the Kalman-filter family and how it relates to the Bayesian filter. In this book, we are covering only the ones in the colored boxes. The complete derivation of the Kalman filter is given below; this provides a coherent framework for all Kalman-filtering implementations. The different filters fall out of the Bayesian models based on assumptions about the model and sensor noise and the linearity or nonlinearity of the measurement and dynamics models. Let us look at the branch that is colored blue. Addi-

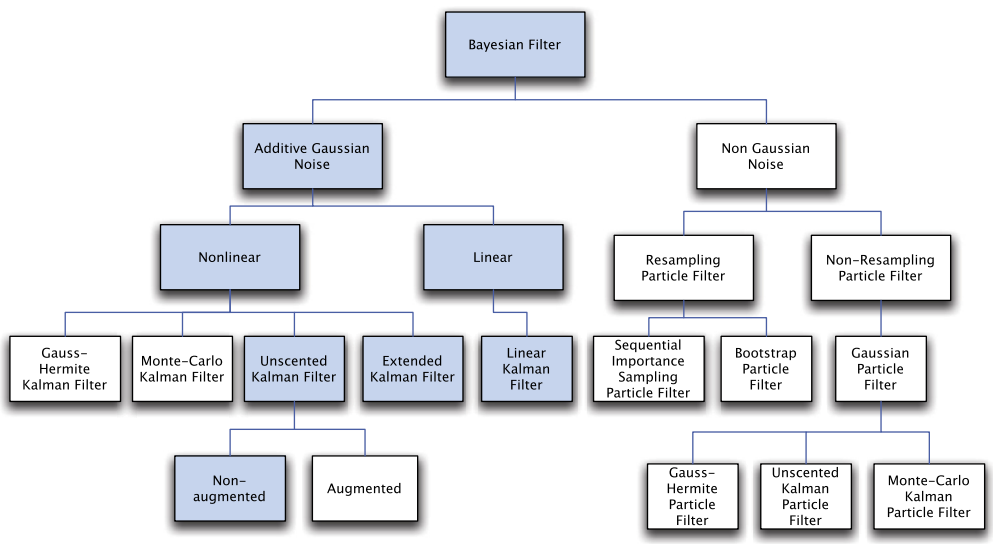


Figure H.1 The Kalman-filter family tree. All are derived from a Bayesian filter. This book covers those in colored boxes.

tive Gaussian noise filters can be linear or nonlinear depending on the type of dynamical and measurement models. In many cases, you can take a nonlinear system and linearize it about the normal operating conditions. You can then use a linear Kalman filter. For example, a spacecraft dynamical model is nonlinear and an Earth sensor that measures the Earth's chordwidth for roll and pitch information is nonlinear. However, if we are only concerned with Earth pointing, and small deviations from nominal pointing, we can linearize both the dynamical equations and measurement equations and use a linear Kalman filter.

If nonlinearities are important we have to use a nonlinear filter. The extended Kalman filter uses partial derivatives of the measurement and dynamical equations. These are computed each time step or with each measurement input. In effect, we are linearizing the system each step and using the linear equations. We do not have to do a linear state propagation, that is propagating the dynamical equations, and could propagate them using numerical integration. If we can get analytical derivatives of the measurement and dynamical equations this is a reasonable approach. If there are singularities in any of the equations this may not work.

The unscented Kalman filter uses the nonlinear equations directly. There are two forms, augmented and nonaugmented. In the former, we created an augmented state vector that includes both the states and the state and measurement noise variables. This may result in better results at the expense of more computation.

All of the filters in this section are Markov, that is the current dynamical state is entirely determined from the previous state. Particle filters are not addressed in this book. They are a class of Monte Carlo methods. Monte Carlo (named after the famous casino) methods are computational algorithms that rely on random sampling to obtain results. For example, a Monte Carlo approach to our oscillator simulation would be to use the MATLAB® function nrandn to generate the accelerations. We would run many tests to verify that our mass moves as expected.

Our derivation will use the notation $\mathrm{N}(\mu, \sigma^2)$ to represent a normal variable. A normal variable is another word for a Gaussian variable. Gaussian means it is distributed as the normal distribution with mean μ (average) and variance σ^2. The following code from Gaussian computes a Gaussian or normal distribution around a mean of 2 for a range of standard deviations. Example H.2 shows a plot. The height of the plot indicates how likely a given measurement of the variable is to have that value. Example H.2 generates a Gaussian distribution.

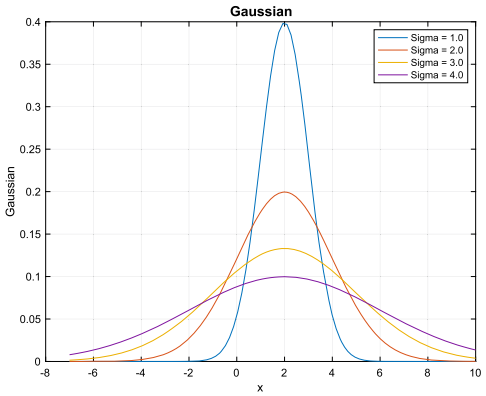
Given the probabilistic state-space model in discrete time [1]

$$x_k = f_k(x_{k-1}, w_{k-1}) \tag{H.42}$$

where x is the state vector and w is the noise vector, the measurement equation is

$$y_k = h_k(x_k, v_n) \tag{H.43}$$

```
mu              = 2;              % Mean
sigma           = [1 2 3 4]; % Standard deviation
n               = length(sigma);
x               = linspace(-7,10);

%% Simulation
xPlot = zeros(n,length(x));
s     = cell(1,n);

for k = 1:length(sigma)
  s{k}        = sprintf('Sigma = %3.1f',sigma(k));
  f           = -(x-mu).^2/(2*sigma(k)^2);
  xPlot(k,:) = exp(f)/sqrt(2*pi*sigma(k)^2);
end

%% Plot the results
Plot2D(x,xPlot,'x','Gaussian','Gaussian')
legend(s)
```



Example H.2: Gaussian distribution.

where v_n is the measurement noise. This has the form of a hidden Markov model (HMM) because the state is hidden.

If the process is Markovian, then the future state x_k is dependent only on the current state x_{k-1} and is not dependent on the past states. This can be expressed in the equation,

$$p(x_k|x_{1:k-1}, y_{1:k-1}) = p(x_k|x_{k-1}) \tag{H.44}$$

The | means given. In this case, the first term is read as "the probability of x_k given $x_{1:k-1}$ and $y_{1:k-1}$". This is the probability of the current state given all past states and all measurements up to the $k-1$ measurement. The past, x_{k-1} is independent of the future given the present,

$$p(x_{k-1}|x_{k:T}, y_{k:T}) = p(x_{k-1}|x_k) \tag{H.45}$$

where T is the last sample and the measurements y_k are conditionally independent given x_k; that is, they can be determined using only x_k and are not dependent on $x_{1:k}$ or $y_{1:k-1}$. This can be expressed as

$$p(y_k|x_{1:k}, y_{1:k-1}) = p(y_k|x_k) \tag{H.46}$$

We can define the recursive Bayesian optimal filter that computes the distribution

$$p(x_k|y_{1:k}) \tag{H.47}$$

given:

- the prior distribution $p(x_0)$, where x_0 is the state prior to the first measurement;
- the state-space model

$$x_k \sim p(x_k|x_{k-1}) \tag{H.48}$$

$$y_k \sim p(y_k|x_k) \tag{H.49}$$

- the measurement sequence $y_{1:k} = y_1, \ldots, y_k$.
  Computation is based on the recursion rule

$$p(x_{k-1}|y_{1:k-1}) \rightarrow p(x_k|y_{1:k}) \tag{H.50}$$

This means we get the current state x_k from the prior state x_{k-1} and all the past measurements $y_{1:k-1}$. Assume we know the posterior distribution of the previous time step

$$p(x_{k-1}|y_{1:k-1}) \tag{H.51}$$

The joint distribution of x_k, x_{k-1} given $y_{1:k-1}$ can be computed as

$$p(x_k, x_{k-1}|y_{1:k-1}) = p(x_k|x_{k-1}, y_{1:k-1})p(x_{k-1}|y_{1:k-1}) \tag{H.52}$$

$$= p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1}) \tag{H.53}$$

because this is a Markov process. Integrating over x_{k-1} gives the prediction step of the optimal filter, which is the Chapman–Kolmogorov equation

$$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1}, y_{1:k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1} \tag{H.54}$$

The Chapman–Kolmogorov equation is an identity relating the joint probability distributions of different sets of coordinates on a stochastic process. The measurement update state is found from the Bayes rule

$$P(x_k|y_{1:k}) = \frac{1}{C_k}p(y_k|x_k)p(x_k|y_{k-1}) \tag{H.55}$$

$$C_k = p(y_k|y_{1:k-1}) = \int p(y_k|x_k)p(x_k|y_{1:k-1})dx_k \tag{H.56}$$

where C_k is the probability of the current measurement, given all past measurements.

If the noise is additive and Gaussian with the state covariance Q_n and the measurement covariance R_n, the model and measurement noise have zero mean, we can write the state equation as

$$x_k = f_k(x_{k-1}) + w_{k-1} \tag{H.57}$$

where x is the state vector and w is the noise vector. The measurement equation becomes

$$y_k = h_k(x_k) + v_n \tag{H.58}$$

Given that Q is not time dependent we can write

$$p(x_k|x_{k-1}, y_{1:k-1}) = \mathrm{N}(x_k; f(x_{k-1}), Q) \tag{H.59}$$

where we recall that N is a normal variable, in this case with mean $x_k; f(x_{k-1})$, which means (x_k given $f(x_{k-1})$ and variance Q. We can now write the prediction step Eq. (H.54) as

$$p(x_k|y_{1:k-1}) = \int \mathrm{N}(x_k; f(x_{k-1}), Q)p(x_{k-1}|y_{1:k-1})dx_{k-1} \tag{H.60}$$

We need to find the first two moments of x_k. A moment is the expected value (or mean) of the variable. The first moment is of the variable, the second is of the variable squared and so forth. They are

$$E[x_k] = \int x_k p(x_k|y_{1:k-1})dx_k \tag{H.61}$$

$$E[x_k x_k^T] = \int x_k x_k^T p(x_k|y_{1:k-1})dx_k \tag{H.62}$$

where E means expected value, $E[x_k]$ is the mean, and $E[x_k x_k^T]$ is the covariance. Expanding the first moment and using the identity $E[x] = \int x\mathrm{N}(x; f(s), \Sigma)dx = f(s)$, where s is any argument,

$$E[x_k] = \int x_k \left[\int \mathrm{d}(x_k; f(x_{k-1}), Q)p(x_{k-1}|y_{1:k-1})dx_{k-1} \right] dx_k \tag{H.63}$$

$$= \int x_k \left[\int \mathrm{N}(x_k; f(x_{k-1}), Q)dx_k \right] p(x_{k-1}|y_{1:k-1})dx_{k-1} \tag{H.64}$$

$$= \int f(x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1} \tag{H.65}$$

Assuming that $p(x_{k-1}|y_{1:k-1}) = \mathrm{N}(x_{k-1}; \hat{x}_{k-1|k-1}, P^{xx}_{k-1|k-1})$ where P^{xx} is the covariance of x and noting that $x_k = f_k(x_{k-1}) + w_{k-1}$ we get

$$\hat{x}_{k|k-1} = \int f(x_{k-1})\mathrm{N}(x_{k-1}; \hat{x}_{k-1|k-1}, P^{xx}_{k-1|k-1})dx_{k-1} \tag{H.66}$$

For the second moment

$$E[x_k x_k^T] = \int x_k x_k^T p(x_k|y_{1:k-1})dx_k \tag{H.67}$$

$$= \int \left[\int \mathrm{N}(x_k; f(x_{k-1}), Q)x_k x_k^T dx_k \right] p(x_{k-1}|y_{1:k-1})dx_{k-1} \tag{H.68}$$

which results in

$$P^{xx}_{k|k-1} = Q + \int f(x_{k-1})f^T(x_{k-1})\mathrm{N}(x_{k-1}; \hat{x}_{k-1|k-1}, P^{xx}_{k-1|k-1})dx_{k-1} - \hat{x}^T_{k|k-1}\hat{x}_{k|k-1} \tag{H.69}$$

The covariance for the initial state is Gaussian and is P^{xx}_0. The Kalman filter can be written without further approximations as

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_n\left[y_k - \hat{y}_{k|k-1}\right] \tag{H.70}$$

$$P^{xx}_{k|k} = P^{xx}_{k|k-1} - K_n P^{yy}_{k|k-1} K_n^T \tag{H.71}$$

$$K_n = P^{xy}_{k|k-1}\left[P^{yy}_{k|k-1}\right]^{-1} \tag{H.72}$$

where K_n is the Kalman gain and P^{yy} is the measurement covariance. The solution of these equations requires the solution of five integrals of the form

$$I = \int g(x)\mathrm{N}(x; \hat{x}, P^{xx})dx \tag{H.73}$$

The three integrals needed by the filter are:

$$P^{yy}_{k|k-1} = R + \int h(x_n)h^T(x_n)\mathrm{N}(x_n; \hat{x}_{k|k-1}, P^{xx}_{k|k-1})dx_k - \hat{x}^T_{k|k-1}\hat{y}_{k|k-1} \tag{H.74}$$

$$P^{xy}_{k|k-1} = \int x_n h^T(x_n)\mathrm{N}(x_n; \hat{x}_{k|k-1}, P^{xx}_{k|k-1})dx \tag{H.75}$$

$$\hat{y}_{k|k-1} = \int h(x_k)\mathrm{N}(x_k; \hat{x}_{k|k-1}, P^{xx}_{k|k-1})dx_k \tag{H.76}$$

Assume we have a model of the form

$$x_k = A_{k-1}x_{k-1} + B_{k-1}u_{k-1} + q_{k-1} \tag{H.77}$$

$$y_k = H_k x_k + r_k \tag{H.78}$$

where

- $x_k \in \Re^n$ is the state of system at time k.
- m_k is the mean state at time k.
- A_{k-1} is the state transition matrix at time $k-1$.
- B_{k-1} is the input matrix at time $k-1$.
- u_{k-1} is the input at time $k-1$.
- $q_{k-1}, \mathrm{N}(0, Q_k)$, is the Gaussian process noise at time $k-1$.
- $y_k \in \Re^m$ is the measurement at time k.
- H_k is the measurement matrix at time k. This is found from the Jacobian (derivatives) of $h(x)$.
- $r_k = \mathrm{N}(0, R_k)$ is the Gaussian measurement noise at time k.
- The prior distribution of the state is $x_0 = \mathrm{N}(m_0, P_0)$ where parameters m_0 and P_0 contain all prior knowledge about the system. m_0 is the mean at time zero and P_0 is the covariance. Since our state is Gaussian this completely describes the state.
- $\hat{x}_{k|k-1}$ is the mean of x at k given $\hat{x}$ at $k-1$.
- $\hat{y}_{k|k-1}$ is the mean of y at k given $\hat{x}$ at $k-1$.

$\Re^n$ means real numbers in a vector of order n, that is the state has n quantities. In probabilistic terms the model is

$$p(x_k|x_{k-1}) = \mathrm{N}(x_k; A_{k-1}x_{k-1}, Q_k) \tag{H.79}$$
$$p(y_k|x_k) = \mathrm{N}(y_k; H_k x_k, R_k) \tag{H.80}$$

The integrals become simple matrix equations. In the following equations P_k^- means the covariance prior to the measurement update.

$$P_{k|k-1}^{yy} = H_k P_k^- H_k^T + R_k \tag{H.81}$$
$$P_{k|k-1}^{xy} = P_k^- H_k^T \tag{H.82}$$
$$P_{k|k-1}^{xx} = A_{k-1} P_{k-1} A_{k-1}^T + Q_{k-1} \tag{H.83}$$
$$\hat{x}_{k|k-1} = m_k^- \tag{H.84}$$
$$\hat{y}_{k|k-1} = H_k m_k^- \tag{H.85}$$

The prediction step becomes

$$m_k^- = A_{k-1} m_{k-1} \tag{H.86}$$
$$P_k^- = A_{k-1} P_{k-1} A_{k-1}^T + Q_{k-1} \tag{H.87}$$

The first term in the above covariance equation propagates the covariance based on the state-transition matrix, A. Q_{k+1} adds to this to form the next covariance. Process noise Q_{k+1} is a measure of the accuracy of the mathematical model, A, in representing the system and includes the effects of unmodeled inputs. For example, suppose A was a mathematical model that damped all states to zero. Without Q, P would go to zero. However, if we were not that certain about the model, the covariance would never be less than Q. Picking Q can be difficult. In a dynamical system with uncertain disturbances, you can compute the standard deviation of the disturbances to compute Q. If the model, A, is uncertain then you might do a statistical analysis of the range of models. Or you can try different Q in simulation and see which ones work best.

The update step is

$$v_k = y_k - H_k m_k^- \tag{H.88}$$
$$S_k = H_k P_k^- H_k^T + R_k \tag{H.89}$$
$$K_k = P_k^- H_k^T S_k^{-1} \tag{H.90}$$
$$m_k = m_k^- + K_k v_k \tag{H.91}$$
$$P_k = P_k^- - K_k S_k K_k^T \tag{H.92}$$

where S_k is an intermediate quantity and v_k is the residual. The residual is the difference between the measurement and your estimate of the measurement given the estimated states. R is the covariance matrix of the measurements. If the noise is not white, a different filter should be used. White noise has equal energy at all frequencies. Many types of noise, such as the noise from an imager, are not white noise but are bandlimited,

that is it has noise in a limited range of frequencies. You can add additional states to A to model the noise better. For example, adding a low-pass filter to the band limits the noise. This makes the size of the A matrix larger requiring more computation.

H.4. Extended Kalman filter

The recursive least-squares method is implemented with an iterated extended Kalman filter measurement update. The state update is

$$\dot{x} = f(x, u, t) \tag{H.93}$$

The covariance update requires the Jacobian, that is the partial of $f(x, u, t)$.

$$A(x, u, t) = \frac{\partial f}{\partial x} \tag{H.94}$$

so that

$$f(x, u, t) \approx A(x, u, t)x \tag{H.95}$$

A is then discretized. This must be done each time step. The partial can only be done if there are no singularities or discontinuities in the right-hand side.

Iteration is applied to the measurement update. It reduces the measurement error due to having an inexact state estimate that would lead to an inexact measurement update. The equation is

$$\begin{aligned} x_{k,i+1} &= x_{k,0} + K_{k,i}\left[y - h(x_{k,i}) - H(x_{k,i})(x_{k,0} - x_{k,i})\right] \\ K_{k,i} &= P_{k,0}H^T(x_{k,i})\left[H(x_{k,i}P_{k,0}H^T(x_{k,i} + R_k\right]^{-1} \\ P_{k,i+1} &= \left[1 - K_{k,i}H(x_{k,i}\right]P_{k,0} \end{aligned} \tag{H.96}$$

where i denotes the iteration. Note that on the right-hand side P and R are always from step k and are not updated during the iteration.

H.5. Unscented Kalman filter

The UKF can achieve greater estimation performance than the extended Kalman filter (EKF) through the use of the unscented transformation (UT). The UT allows the UKF to capture first- and second-order terms of the nonlinear system [2]. The state-estimation algorithm employed is that given by van der Merwe [3] and the parameter-estimation algorithm given by [2]. Unlike the EKF, the UKF does not require any derivatives or Jacobians of either the state equations or measurement equations. Instead of just propagating the state, the filter propagates the state and additional sigma points that are the states plus the square roots of rows or columns of the covariance matrix.

Thus the state and the state plus a standard deviation are propagated. This captures the uncertainty in the state. It is not necessary to numerically integrate the covariance matrix. No specialized, linearized or simplified simulation is needed. Thus the algorithms can update the actual simulations.

The UKF has three parts:

1. Initialization;
2. UKF prediction;
3. UKF update.

The weights are computed once at the initialization of the filter [4]

$$W_m^0 = \frac{\lambda}{n+\lambda} \tag{H.97}$$

$$W_c^0 = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta \tag{H.98}$$

$$W_m^i = W_i^c = \frac{1}{2(n+\lambda)} \tag{H.99}$$

$$W_c^i = W_m^i \tag{H.100}$$

where

$$\lambda = \alpha^2(n+\kappa) - n \tag{H.101}$$

$$\gamma = \sqrt{L+\lambda} \tag{H.102}$$

The constant α determines the spread of the sigma points around $\hat{X}$ and is usually set to between 10^{-4} and 1. β incorporates prior knowledge of the distribution of x and is 2 for a Gaussian distribution. κ is set to 0 for state estimation.

$$c = \alpha^2(n+\kappa) \tag{H.103}$$

$$w_m = \begin{bmatrix} W_m^0 & \cdots & W_m^{2n} \end{bmatrix}^T \tag{H.104}$$

$$W = \left(I - \begin{bmatrix} w_m & \cdots & w_m \end{bmatrix}\right) \times \begin{bmatrix} W_c^0 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & W_c^{2n} \end{bmatrix} \times \left(I - \begin{bmatrix} w_m & \cdots & w_m \end{bmatrix}\right)^T \tag{H.105}$$

where I is the identity matrix with dimensions $2n \times 2n$.

H.6. UKF state-prediction step

The prediction step is as follows

$$X_{k-1} = \begin{bmatrix} m_{k-1} & m_{k-1} & m_{k-1} \end{bmatrix} + \sqrt{c}\begin{bmatrix} 0 & \sqrt{P_{k-1}} & -\sqrt{P_{k-1}} \end{bmatrix} \tag{H.106}$$

$$\hat{X}_k = f(X_{k-1}, k-1) \tag{H.107}$$
$$m_k^- = \hat{X}_k w_m \tag{H.108}$$
$$P_k^- = \hat{X}_k W \hat{X}_k^T + Q_{k-1} \tag{H.109}$$

where m is the mean state and Q_{k-1} is the model covariance matrix. The square root of P can be found in numerous ways. One is the matrix square root. A more efficient method is to take the Cholesky decomposition of P, which is an approximation of the matrix square root.

The measurement update step is

$$X_{k-1}^- = \begin{bmatrix} m_{k-1}^- & m_{k-1}^- & m_{k-1}^- \end{bmatrix} + \sqrt{c}\begin{bmatrix} 0 & \sqrt{P_{k-1}^-} & -\sqrt{P_{k-1}^-} \end{bmatrix} \tag{H.110}$$
$$Y_k^- = h(X_{k-1}^-, k) \tag{H.111}$$
$$\mu_k = Y_k^- w_m \tag{H.112}$$
$$S_k = Y_k^- W \left[Y_k^-\right]^T + R_k \tag{H.113}$$
$$C_k = X_k^- W \left[Y_k^-\right]^T \tag{H.114}$$

where R_k is the measurement noise covariance matrix. Compute the gain K_k, mean state m_k, and covariance P_k,

$$K_k = C_k S_k^- 1 \tag{H.115}$$
$$m_k = m_k^- + K_k\left[y_k - \mu_k\right] \tag{H.116}$$
$$P_k = P_k^- - K_k S_k K_k^T \tag{H.117}$$

H.7. Kalman-filter example

This section provides an example of a Kalman filter with a nonlinear measurement. This highlights the difference between the different types of filters.

H.7.1 Dynamical and measurement model

This example is for a nonlinear spring with a nonlinear measurement. The model is shown in Fig. H.2.

The spring has a cubic spring in parallel with a linear spring. The equations of motion and the measurement equation are

$$\dot{x} = v \tag{H.118}$$
$$m\dot{v} = -cv - k_l x - k_c x^3 + f \tag{H.119}$$
$$\theta = \tan^{-1}\left(\frac{x}{w}\right) \tag{H.120}$$

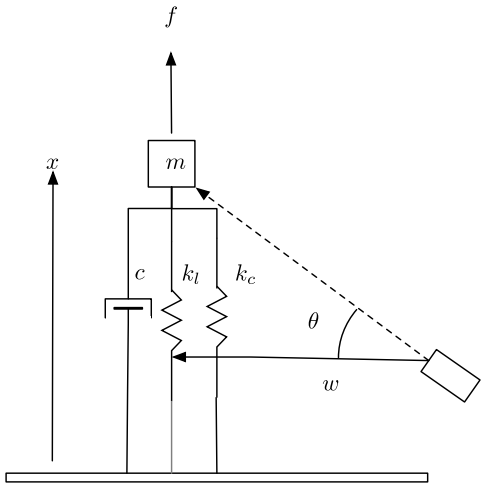


Figure H.2 Mass with a 3rd-order spring in parallel with a damper and linear spring.

The linearized equations, about $x = 0$ are

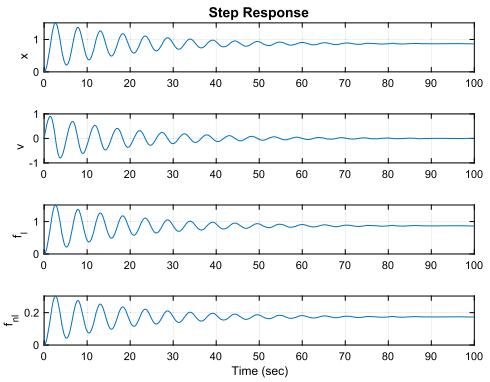
$$\dot{x} = v \tag{H.121}$$

$$m\dot{v} = -cv - k_l x + f \tag{H.122}$$

$$\theta = \frac{x}{w} \tag{H.123}$$

The dynamical model is simulated in Example H.3.

```
d           = RHSNLSpring;
d.f         = 1;
n           = 1000;
dT          = 0.1;
xP          = zeros(4,n);
x           = [0;0];
for k = 1:n
   xP(:,k) = [x;d.kL*x(1);d.kC*x(1)];
   x = RK4(@RHSNLSpring,x,dT,0,d);
end
[t,tL] = TimeLabl((0:n-1)*dT);
Plot2D(t,xP,tL,{'x' 'v' 'f_l' 'f_{nl}'},'Step␣
       Response')
```

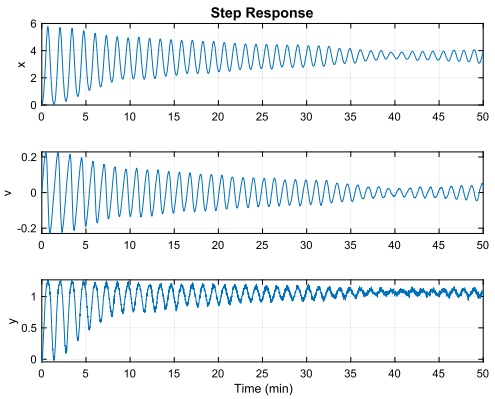


Example H.3: Nonlinear-spring simulation. It shows the nonlinear and linear spring forces.

Example H.4 runs the model once and saves the state and measurement data into a mat file for use in the Kalman-filter demonstrations. The noise obscures the mea-

surement once the position has been damped. The force noise also is visible once the damping has had an effect. The 1σ noise values are in the mat file so that the Kalman filters can set their covariances properly.

```
d             = RHSNLSpring;
f             = 0.01;
d.c           = 2e-3; % Lessen the damping
d.kL          = 1e-4; % Lower the spring constant
d.kC          = 2e-4; % Lower the spring constant
      make it twice the linear spring
noiseForce    = 1e-3; % 1 sigma acceleration noise
noiseMeas     = 0.02; % 1 sigma acceleration noise
n             = 3000; % Number of time steps
dT            = 1;    % Time step, seconds
xP            = zeros(3,n);
x             = [0;0];
w             = 2; % Baseline

for k = 1:n
  y           = atan(x(1)/w) + noiseMeas*randn;
  xP(:,k)     = [x;y];
  d.f         = f + noiseForce*randn;
  x           = RK4(@RHSNLSpring,x,dT,0,d);
end
[t,tL] = TimeLabl((0:n-1)*dT);
Plot2D(t,xP,tL,{'x' 'v' 'y'},'Step Response')

save('KFSim','xP', 'noiseMeas', 'noiseForce', 'd'
       , 'dT', 'n', 'w');
```



Example H.4: Nonlinear-spring simulation generating noisy measurements.

H.7.2 Linear Kalman filter

The first script, KFNLSpring, demonstrates the linear Kalman filter. For the linearized Kalman filter we need the linearized versions of the state and measurement equations around a position x_t. These are found by taking the derivatives of the equations about that point. Given

$$\dot{x} = g(x, u, t) \tag{H.124}$$

$$y = h(x, t) \tag{H.125}$$

The linearized versions at time t are

$$\Phi = \frac{\partial g(x, u, t)}{\partial x}\Big|_t \tag{H.126}$$

$$H = \frac{\partial h(x, t)}{\partial x}\Big|_t \tag{H.127}$$

The derivative of the measurement is

$$\frac{\partial \tan^{-1}\left(\frac{x}{w}\right)}{\partial x} = \frac{w}{w^2 + x^2} \tag{H.128}$$

For $x = 0$

$$H = \begin{bmatrix} \frac{1}{w} & 0 \end{bmatrix} \tag{H.129}$$

the state matrix is

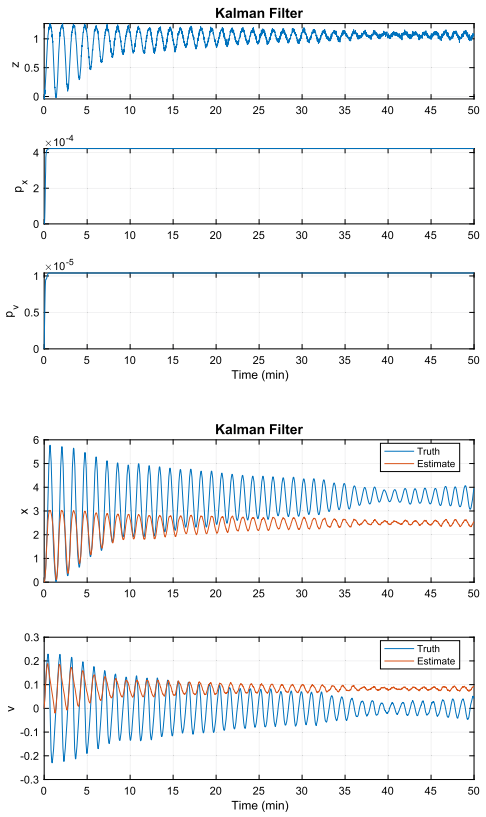
$$a = \begin{bmatrix} 0 & 1 \\ -k_l & -c \end{bmatrix} \tag{H.130}$$

and the input matrix is

$$b = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{H.131}$$

The Kalman-filter code has two parts, measurement update, and state propagation. q is a scalar. e is a 2-by-2 identify matrix. Example H.5 shows the results.

```
s             = load('KFSim');
n             = s.n; % Number of steps
r             = s.noiseMeas^2; % Measurement noise
q             = s.noiseForce^2; % Plant noise
h             = [1/s.w 0]; % Measurement matrix
a             = [0 1;-s.d.kL/s.d.m -s.d.c/s.d.m]; %
        State transition matrix
b             = [0; 1/s.d.m]; % Input matrix

% Discretize
[phi,gamma] = C2DZOH(a,b,s.dT);
xE            = [0;0]; % Estimated state
u             = s.d.f; % Input
p             = [0 0;0 0]; % Initial covariance
xP            = zeros(7,n);
e             = eye(2);

%% Run the Kalman Filter
for j = 1:n

  % Measurement
  z = s.xP(3,j);

  % Store variables to plot
  xP(:,j) = [z;diag(p);s.xP(1:2,j);xE];

  % Kalman Filter measurement update
  k  = p*h'/(h*p*h' + r);
  xE = xE + k*(z - h*xE);
  p  = (e - k*h)*p;

  % Kalman Filter state update
  xE = phi*xE + gamma*u;
  p  = phi*p*phi' + gamma'*q*gamma;

end

%% Plot
yL = {'z' 'p_x' 'p_v' 'x' 'v' };
lL = {{'Truth','Estimate'} {'Truth','Estimate'}};

[t,tL] = TimeLabl(0:(n-1)*s.dT);
Plot2D(t,xP(1:3,:),tL,yL(1:3),'Kalman Filter');
Plot2D(t,xP(4:7,:),tL,yL(4:5),'Kalman Filter',...
  'lin',{'[1 3]' '[2 4]'},[],[],[],[],lL);
```



Example H.5: Conventional (or linear) Kalman filter for a nonlinear spring.

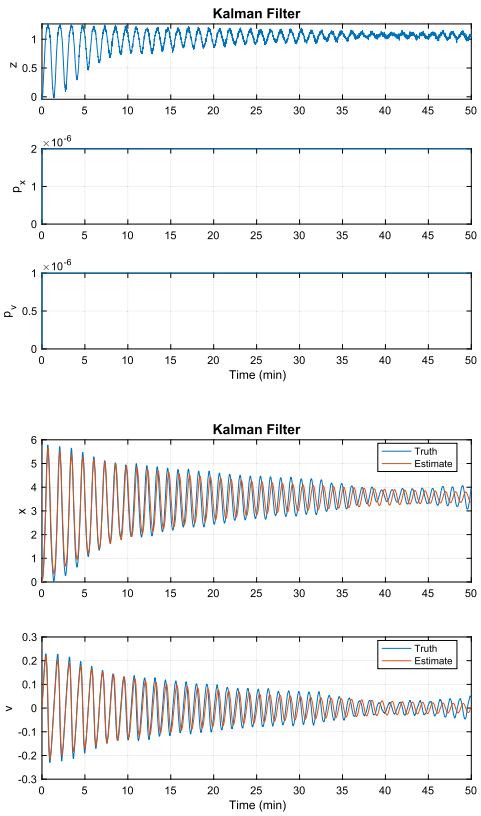
H.7.3 Extended Kalman filter

Example H.6, demonstrates the extended Kalman filter. This requires the functions

1. RHSPartialNLSpring for the partial derivative of the right-hand side;
2. MeasNLSpring for the measurement;
3. MeasPartialNLSpring for the partial derivative of the measurement.

In this case, the partials can be computed analytically. In many cases, these must be done numerically. The results are shown in Example H.6. The estimates track the truth values well. The measurements do come in and out of phase at times.

```
s             = load('KFSim');
n             = s.n; % Number of steps
r             = s.noiseMeas^2; % Measurement noise
q             = s.noiseForce^2; % Plant noise
xE            = [0;0]; % Estimated state
p             = [0 0;0 0]; % Initial covariance
xP            = zeros(7,n);
s.d.w         = s.w;

% Initialize the filter
dEKF          = KFInitialize( 'ekf','f',
        @RHSNLSpring,...
'dT',s.dT,...
'fData',s.d,'h',@MeasNLSpring,...
'hX',@MeasPartialNLSpring,'hData',s.d,...
'fX',@RHSPartialNLSpring,'p',p,...
'q',q,'x',xE,'m',xE,'r',r);

%% EKF Estimation Loop
t = 0;
for k = 1:n
  % Measurement
  z = s.xP(3,k);

  % Store variables to plot
  xP(:,k)         = [z;diag(dEKF.p);s.xP(1:2,k);
          dEKF.m];

        % Extended Kalman Filter
  dEKF.t          = t;
        dEKF.y          = z;
        dEKF            = EKFPredict( dEKF );
        dEKF            = EKFUpdate( dEKF );
  t               = t + s.dT;
end

%% Plot
yL = {'z' 'p_x' 'p_v' 'x' 'v' };
lL = {{'Truth','Estimate'} {'Truth','Estimate'}};

[t,tL] = TimeLabl(0:(n-1)*s.dT);
Plot2D(t,xP(1:3,:),tL,yL(1:3),'Kalman Filter');
Plot2D(t,xP(4:7,:),tL,yL(4:5),'Kalman Filter',...
  'lin',{'[1 3]' '[2 4]'},[],[],[],[],lL);


%-------------------------------------
% $Date$
% $Revision$
```

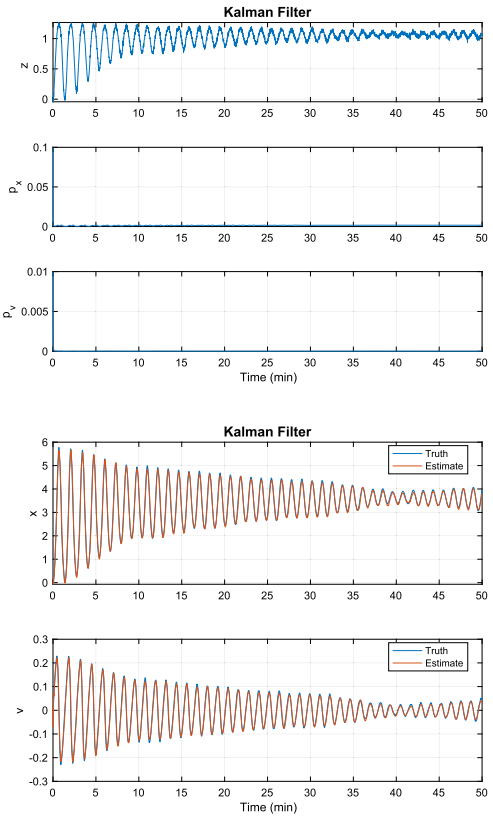


Example H.6: Extended Kalman filter for a nonlinear spring.

H.7.4 Unscented Kalman filter

The script, Example H.7, demonstrates the unscented Kalman filter. It uses the same functions as the simulation. The UKF tracks the pendulum better than the EKF.

```
s             = load('KFSim');
n             = s.n; % Number of steps
r             = s.noiseMeas^2; % Measurement noise
q             = s.noiseForce^2; % Plant noise
xE            = [0;0]; % Estimated state
p             = [0.1 0;0 0.01]; % Initial
        covariance
xP            = zeros(7,n);
s.d.w         = s.w;

% Initialize the filter
dUKF = KFInitialize( 'ukf','f',@RHSNLSpring,...
'dT',s.dT,...
'fData',s.d,'h',@MeasNLSpring,...
'hData',s.d,'p',p,...
'q',q,'x',xE,'m',xE,'r',r);
% Get the UKF weights
dUKF          = UKFWeight( dUKF);

%% UKF Estimation Loop
t = 0;
for k = 1:n
  % Measurement
  z         = s.xP(3,k);

  % Store variables to plot
  xP(:,k) = [z;diag(dUKF.p);s.xP(1:2,k);dUKF.m];

        % Add a measurement source
  dUKF.t                  = t;
        dUKF.y.data             = z;
        dUKF.y.param.hFun           = @MeasNLSpring
                ;
        dUKF.y.param.hData        = s.d;
        dUKF.y.param.r            = r;

  % Unscented Kalman Filter
        dUKF = UKFPredict( dUKF );
        dUKF = UKFUpdate( dUKF );
  t     = t + s.dT;
end

%% Plot
yL = {'z' 'p_x' 'p_v' 'x' 'v' };
lL = {{'Truth','Estimate'} {'Truth','Estimate'}};

[t,tL] = TimeLabl(0:(n-1)*s.dT);
Plot2D(t,xP(1:3,:),tL,yL(1:3),'Kalman Filter');
Plot2D(t,xP(4:7,:),tL,yL(4:5),'Kalman Filter',...
  'lin',{'[1 3]' '[2 4]'},[],[],[],[],lL);
```



Example H.7: Unscented Kalman filter for a nonlinear spring.

References

[1] S. Sarkka, Lecture 3: Bayesian Optimal Filtering Equations and the Kalman Filter, Tech. Rep., Department of Biomedical Engineering and Computational Science, Aalto University School of Science, February 2011.

[2] M.C. VanDyke, J.L. Schwartz, C.D. Hall, Unscented Kalman filtering for spacecraft attitude state and parameter estimation, in: Proceedings of the AAS/AIAA Space Flight Mechanics Conference, 2005, AAS 04-115.

[3] R. van der Merwe, E.A. Wan, The square-root unscented Kalman filter for state and parameter-estimation, in: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, 2001, pp. 3461–3464.
[4] J. Hartikainen, A. Solin, S. Särkkä, Optimal Filtering with Kalman Filters and Smoothers a Manual for the Matlab toolbox EKF/UKF Version 1.3, Tech. Rep., Aslto University, August 2011.

APPENDIX I

Orbit theory

I.1. Space story

The author's wife came to watch one of the GSTAR III orbit-raising maneuvers. We did over 160 maneuvers over the course of the recovery. Each maneuver meant turning on the Electrothermal Hydrazine Thrusters (EHTs). The maneuver involved closing the pitch loop, heating up the thrusters and turning them on for the burn duration. We then shut everything down returning one orbit period later. I asked her what she thought. She said, "It was boring."

I.2. Introduction

This chapter discusses orbit representation and dynamics. It provides a brief introduction to orbit geometry followed by a discussion of orbit representations. Many spacecraft attitude disturbances are functions of the orbital position. For Earth- or planet-pointing spacecraft, such as low-Earth orbit constellations, it is necessary to know the orbital position to determine the pointing direction. Formation flying is another area of interest and in formation flying relative orbital motion is tied to the individual satellite pointing. In some cases the attitude control engineer can work with a precomputed orbital track. In other cases, it is necessary to numerically integrate the orbit along with the attitude.

I.3. Representations of orbits

I.3.1 Orbital geometry

An orbit involves at least two bodies, for example a planet and a spacecraft. In the ideal two-body case the two bodies rotate about the common center-of-mass, known as the barycenter. For all practical spacecraft cases, the spacecraft mass itself is negligible and this means that the satellite follows a conic section path about the primary body's center-of-mass. The orbit may be elliptical, with an eccentricity less than one, parabolic with an eccentricity equal to one or hyperbolic with an eccentricity greater than one. Elliptical orbits stay around the orbit center. Parabolic and hyperbolic orbits go to infinity. Fig. I.1 shows the geometry of an elliptical orbit. This is a planar orbit in which the orbital motion is two-dimensional.

The semimajor axis a is

$$a = \frac{r_a + r_p}{2} \tag{I.1}$$

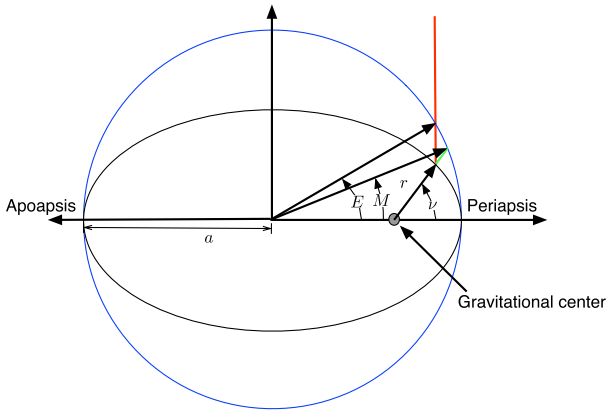


Figure I.1 Elliptical orbit.

where r_a is the apoapsis (apogee for Earth) radius, or point furthest from the central planet, and r_p is the periapsis radius (perigee for the Earth), or closest point to the planet. The eccentricity, e, of the orbit is

$$e = \frac{\frac{r_a}{r_p} - 1}{\frac{r_a}{r_p} + 1} \tag{I.2}$$

When $r_a = r_p$ the orbit is circular and $e = 0$. This formula is not meaningful for parabolic or hyperbolic orbits. Fig. I.1 shows three angular measurements, M is the mean anomaly, E is the eccentric anomaly, and ν is the true anomaly. All are measured from periapsis. The mean anomaly is related to the mean orbit rate n through a linear function of time and orbit rate

$$M = M_0 + n(t - t_0) \tag{I.3}$$

where n is the orbit rate and t_0 is the time when $M = M_0$. The eccentric anomaly is the angle to the current position as projected onto the ellipse's circumscribing circle, drawn in blue. It is related to the mean anomaly by Kepler's equation

$$M = E - e \sin E \tag{I.4}$$

where e is the orbit eccentricity. This equation needs to be solved numerically in general, but for small values of e this approximation can be used:

$$E \approx M + e \sin M + \frac{1}{2} e^2 \sin 2M \tag{I.5}$$

Higher-order approximations can also be found. The true anomaly is related to the eccentric anomaly through the equation

$$\tan\frac{\nu}{2} = \sqrt{\frac{1+e}{1-e}}\tan\frac{E}{2} \tag{I.6}$$

where ν is the true anomaly. Finally, the orbit radius is

$$r = \frac{a(1-e^2)}{1+e\cos\nu} \tag{I.7}$$

If $e > 1$ in this equation r will go to ∞, as is expected for hyperbolic orbits. For a hyperbolic orbit, a is negative.

I.3.2 Cartesian coordinates

The central-body equations in Cartesian coordinates are

$$\ddot{r} + \mu\frac{r}{|r|^3} = a \tag{I.8}$$

where r is the position vector from the mass center of the central body, a is the acceleration vector of all other disturbances (such as thrusters, solar pressure, and gravitational accelerations due to other planets and planetary asymmetries), and μ is the gravitational parameter for the central body. The radius vector is

$$r = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{I.9}$$

and the radius magnitude is $|r| = \sqrt{x^2+y^2+z^2}$. The equations are nonlinear. However, when the disturbance acceleration a is zero, r traces ellipses, parabolas, or hyperbolas. Equations can also be written in cylindrical or spherical coordinates. a also includes accelerations due to harmonics of the planet's gravity and other nonideal gravitational effects.

I.3.3 Keplerian elements

Seven parameters are necessary to define an orbit of a spacecraft about a spherically symmetric body. One is the gravitational parameter, generally denoted by the symbol μ. There are many ways of representing the other six elements. The most popular are position and velocity (r and ν) vectors and Keplerian elements, both are shown in Fig. I.2.

The Keplerian elements are defined as follows: Two elements define the elliptical orbit. The size of the orbit is determined by the semimajor axis a, which is the average of the perigee radius and apogee radius. The shape of the orbit is defined by the

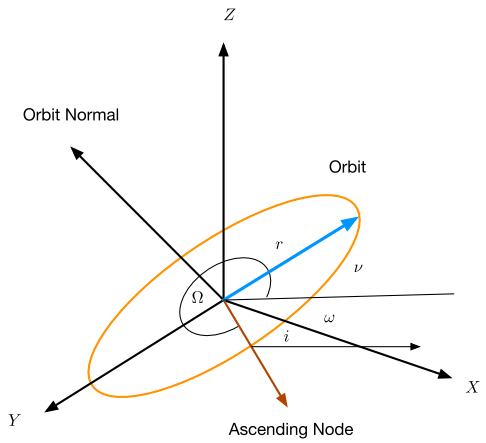


Figure I.2 Orbital elements.

eccentricity, e. Two elements define the orbital plane. Ω is the longitude or right ascension of the ascending node, or the angle from the $+X$-axis of the reference frame to the line where the orbit plane intersects the xy-plane. i is the inclination and is the angle between the xy-plane and the orbit plane. ω is the argument of perigee and is the angle in the orbit plane between the ascending node line and perigee (where the orbit is closest to the center of the central body). ν is the true anomaly and is the angle between perigee and the spacecraft. The mean anomaly M may be used in the element set instead of ν. M or ν tell us where the spacecraft is in its orbit. To summarize, the Keplerian elements are

$$x = \begin{bmatrix} a \\ i \\ \Omega \\ \omega \\ e \\ M \end{bmatrix} \tag{I.10}$$

The variational equations for Keplerian elements are

$$p = a(1 - e^2) \tag{I.11}$$

$$r = \frac{p}{1 + e\cos\nu} \tag{I.12}$$

$$h = \sqrt{\mu p} \tag{I.13}$$

$$b = \sqrt{pa} \tag{I.14}$$

$$n = \frac{h}{ab} \tag{I.15}$$

where h is angular momentum, p is the parameter, i is the inclination, ν is the true anomaly, M is the mean anomal, and n is the orbital rate. The accelerations are a_R, radial, a_T, tangential, and a_N normal to the orbit.

$$\dot{a} = \frac{2a^2(e\sin\nu a_R + \frac{p}{r}a_T)}{h} \tag{I.16}$$

$$\dot{i} = \frac{r\cos(\omega+\nu)a_N}{h} \tag{I.17}$$

$$\dot{\Omega} = \frac{r\sin(\omega+\nu)a_N}{h\sin i} \tag{I.18}$$

$$\dot{\omega} = \frac{\frac{-p\cos\nu a_R + (p+r)\sin\nu a_T}{e} - \frac{r\sin(\omega+\nu)\cos i a_N}{\sin i}}{h} \tag{I.19}$$

$$\dot{e} = \frac{p\sin\nu a_R + (p+r)\cos\nu + re)a_T}{h} \tag{I.20}$$

$$\dot{M} = n + b\frac{p(\cos\nu - 2re)a_R - (p+r)\sin\nu a_T}{ahe} \tag{I.21}$$

For near-circular orbits, $e = 0$ so $p = a$ and $r = a$. We are interested only in the first three equations that become

$$\dot{a} = \frac{2a^{3/2}a_T}{\sqrt{\mu}} \tag{I.22}$$

$$\dot{i} = \sqrt{\frac{a}{\mu}}\cos(\omega+\nu)a_N \tag{I.23}$$

$$\dot{\Omega} = \sqrt{\frac{a}{\mu}}\left(\frac{\sin(\omega+\nu)}{\sin i}\right)a_N \tag{I.24}$$

I.3.4 Equinoctial elements

Modified equinoctial coordinates are a form of orbital elements that are not singular when e is zero nor i is zero. The modified coordinates are

$$\begin{aligned} p &= a(1-e^2) \\ f &= e\cos(\omega+\Omega) \\ g &= e\sin(\omega+\Omega) \\ h &= \tan(i/2)\cos\Omega \\ k &= \tan(i/2)\sin\Omega \\ L &= \Omega+\omega+\nu \end{aligned} \tag{I.25}$$

where p is the semiparameter, a is the semimajor axis, e is the orbital eccentricity, ω is the argument of perigee, Ω is the right ascension of the ascending node, L is the true longitude, and ν is the true anomaly.

The dimensional dynamical equations are

$$\dot{x} = Ga + b \tag{I.26}$$

where a is the acceleration vector with components in the radial, tangential, and normal directions and includes all accelerations except for point-mass gravity, and b drives the true longitude, L.

The state vector is

$$x = \begin{bmatrix} p \\ f \\ g \\ h \\ k \\ L \end{bmatrix} \tag{I.27}$$

and

$$G = \begin{bmatrix} 0 & 2r\omega & 0 \\ \omega s & \omega(\gamma c + f)/q & -\Omega g \\ -\omega c & \omega(\gamma s + g)/q & -\Omega f \\ 0 & 0 & \zeta c \\ 0 & 0 & \zeta s \\ 0 & 0 & \Omega \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \sqrt{\frac{\mu p}{r^2}} \end{bmatrix} \tag{I.28}$$

where

$$r = \frac{p}{q} \tag{I.29}$$

$$c = \cos L \tag{I.30}$$

$$s = \sin L \tag{I.31}$$

$$z = hs - kc \tag{I.32}$$

$$\omega = \sqrt{\frac{p}{\mu}} \tag{I.33}$$

$$q = 1 + fc + gs \tag{I.34}$$

$$\gamma = q + 1 \tag{I.35}$$

$$\Omega = \omega \frac{z}{q} \tag{I.36}$$

$$\zeta = \frac{1}{2}\frac{\omega}{q}(1 + \sqrt{h^2 + k^2}) \tag{I.37}$$

If the accelerations are zero, only L changes. The equinoctial acceleration frame shown in Fig. I.3 shows the definition of the acceleration vector a that multiplies G.

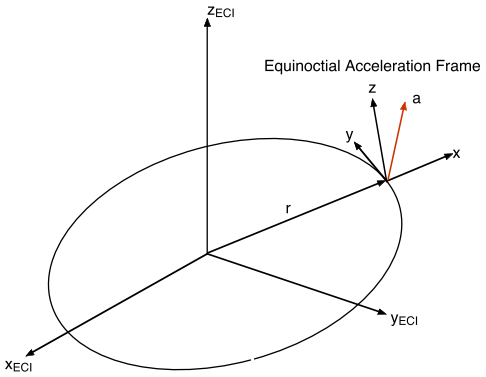


Figure I.3 Equinoctial acceleration frame.

We define the unit vectors as

$$\hat{x} = \frac{r}{|r|} \tag{I.38}$$

$$\hat{y} = \frac{n \times s}{|n \times s|} \tag{I.39}$$

$$\hat{z} = n \tag{I.40}$$

where

$$n = \frac{r \times v}{|r \times v|} \tag{I.41}$$

The transformation matrix from inertial to equinoctial is

$$C_{ITOE} = \begin{bmatrix} \hat{x}^T \\ \hat{y}^T \\ \hat{z}^T \end{bmatrix} \tag{I.42}$$

I.4. Propagating orbits

I.4.1 Introduction

There are four general methods for propagating an orbit. One is Kepler propagation. For example, if there are no perturbing accelerations on the spacecraft (such as thruster firings, solar pressure, or the gravitational pull of other planets) it is possible to just propagate the elements directly. More complex propagators can account for some kinds of perturbations.

The second method is an analytical theory. This is used for planets and their moons. Very complex nonlinear expressions can be used to predict the position of a planet or

moon with high accuracy over long periods of time. Some of these analytical theories are hundreds of pages long. While rarely used for attitude or orbit control work, they can be useful onboard flight computers if it is necessary to predict the position of an astronomical body.

The third method is numerical integration, which is used for most practical orbit modeling. It allows the engineer to incorporate all perturbations in a straightforward fashion and permits easy modification of the models for additional disturbance sources.

The fourth method uses what are known as two-line elements. These are sometimes called NORAD two-line elements. They are orbital fits that include perturbing accelerations like drag, solar pressure, and the higher harmonics of the Earth's gravity. Two-line elements are available for all orbiting spacecraft.

I.4.2 Kepler propagation

If perturbing accelerations are small, it is possible to propagate the orbital elements directly. Kepler propagation is very useful when you just need to test a control system as the spacecraft orbits the planet and you are not concerned with what is happening to the orbit as a result of thruster firings, etc. To determine the orbit it is necessary to know the eccentric anomaly. The equation that must be solved for obtaining the eccentric anomaly from elliptical orbits is

$$M = E - e\sin E \tag{I.43}$$

where e is eccentricity and E is the eccentric anomaly, which is known as Kepler's equation. It is necessary to solve for E iteratively but this can be done in a straightforward fashion. Impulse changes in velocity can be accommodated easily. To change the orbit with an impulse change in velocity in which the velocity changes while the position stays fixed just:

1. Convert the orbital elements to r and v vectors;
2. Add the Δv to v;
3. Convert $v + \Delta v$ and r back to orbital elements.

This procedure is useful for modeling solid-rocket burns or any Δv that happens in a time much less than the orbit period.

I.4.3 Numerical integration

The state equations for orbit propagation are

$$\dot{v} = -\mu\frac{r}{|r|^3} + g(r) + f(\theta) + h \tag{I.44}$$

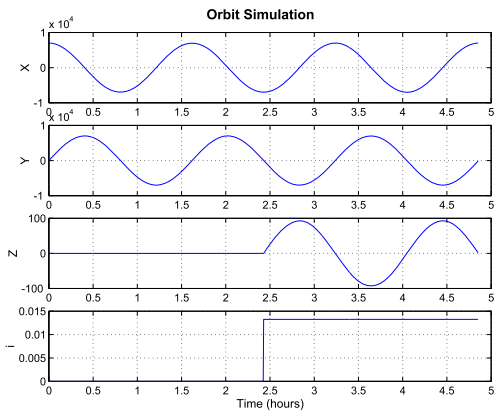
$$\dot{x} = v \tag{I.45}$$

The terms on the right-hand side of the velocity-derivative equation are the spherical gravity acceleration, additional accelerations g that are functions of position, accelerations f that are functions of angles, and accelerations h that are independent of orientation or position (such as thruster firings). Orientation-dependent forces include aerodynamic drag and solar-pressure forces. The orientation may just be the orientation of the spacecraft with respect to inertial space, or may include the relative orientation of components such as solar arrays. If the orientation forces are small, then the orbit equations may be decoupled from the attitude equations and propagated separately. Otherwise, they must be integrated together. For example, the drag on the Space Shuttle in low-Earth orbit is dependent on orientation so the attitude and orbit equations cannot be decoupled.

For precise orbit predictions it is necessary to use integrators with error correction. The most widely used today are the Runge–Kutta $n/n+1$-order integrators in which coefficients of the equations are chosen so that with minimal extra computation a nth-order and $n+1$th-order solution are computed. The results can be compared and the step size adjusted. Some integrators will also change order to minimize the solution error.

The following example shows an orbit simulation which uses a fourth-order Runge–Kutta integrator with the Cartesian orbit equations. Example I.1 shows a simulation of a circular orbit. In this case we suddenly change the z velocity from 0 to 0.1 km/s half-way through the simulation. This causes an instantaneous change in inclination.

```
nSim        = 1000;
xPlot       = zeros(4,nSim);
[r,v]       = El2RV( [7000 0 0 0 0 0 ] );
tEnd        = 3*Period(7000);
dT          = tEnd/nSim;
x           = [r;v];
for k = 1:nSim
  el          = RV2El( x(1:3), x(4:6) );
  xPlot(:,k) = [x(1:3);el(2)];
  x           = RK4( 'FOrbCart', x, dT, 0 );
  if( k == 500 )
    x(6) = 0.1;
  end
end
[t,c] = TimeLabl( (0:(nSim-1))*dT );
Plot2D( t, xPlot, c, ['X';'Y';'Z';'i'], 'Orbit␣
      Simulation' )
PrintFig(1,1,1,'OrbitSim')
```



Example I.1: Orbit simulation of impulsive inclination change.

El2RV converts elements in an array $[a, i, w, W, e, M]$ to r and v. Period returns the period of the orbit given the semimajor axis. RV2El computes r and v from the orbital elements. RK4 is the 4th-order Runge–Kutta integrator. It does not perform step-size adjustment. When $k = 500$ we instantaneously change the z-velocity. This is a

good approximation to an impulsive velocity change, which you might get with a solid rocket motor firing.

x, y, and z all look like harmonic oscillators, as shown above. The inclination change is evident in the plot. Many types of mission planning can be done as simply as the example shown here.

I.5. Gravitational acceleration

I.5.1 Point masses

The simplest possible model for the gravitational acceleration due to a planet is to assume that the planet is a point mass. That means it has no dimensions but finite mass. In this case the acceleration due to a planet on a second body is

$$a = -\mu \frac{r}{|r|^3} \tag{I.46}$$

where r is the vector from the barycenter.

The planet has finite dimensions and since orbits are always outside the planet, the singularity at $|r| = 0$ is not important. The above equation is written with respect to a coordinate frame that is inertially fixed. Since the gravitational acceleration depends only on the radius the latitude and longitude of the subsatellite point (the point on the r vector that is on the planet's surface) is not important. This will not be true when planetary asymmetries are accounted for later.

Assume that there are n gravitational sources, all point masses, then the equations of motion for the satellite can be written as follows.

$$a = -\mu \frac{r}{|r|^3} - G \sum_{j=3}^{N} m_j \left(\frac{d_j}{|d_j|^3} + \frac{\rho_j}{|\rho_j|^3} \right) \tag{I.47}$$

$$\mu = G(m_1 + m_2)$$

where the variables are described in Fig. I.4. The primary planet is the object that produces the largest gravitational acceleration on the spacecraft. It is body 1 and the spacecraft is body 2.

The above equations are important for all Earth-orbiting satellites. For Earth satellites the Sun and Moon generate significant perturbations in the orbit and must be included. For lunar orbits, the Earth and the Sun should be included.

I.5.2 Planetary asymmetries

There are many ways to deal with asymmetries in planets. One would be to model the planet as a set of point masses contained within the surface of the planet. The second

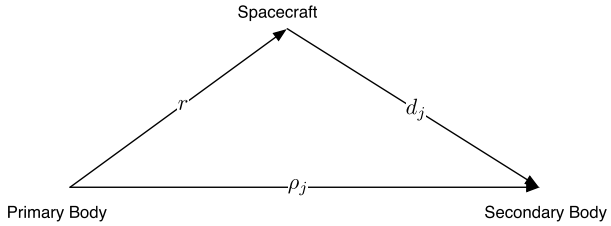


Figure I.4 Position vectors in the *n*-body system.

is to expand the gravitational potential in a spherical-harmonic expansion and compute the gravity forces from the gradient of the potential. Other expansions could also be used.

The spherical-harmonic expansion method works as follows. Define the perturbing gravitational potential of a planet as

$$V = \frac{\mu}{r}\sum_{n=2}^{\infty}\left[\left(\frac{a}{r}\right)^n \sum_{m=0}^{\infty}\left(S_{n,m}\sin m\lambda + C_{n,m}\cos m\lambda\right)P_{n,m}(\sin\phi)\right] \tag{I.48}$$

defined in the planet fixed frame. a is the radius of the planet. The definition of the coordinates is shown in Fig. I.5.

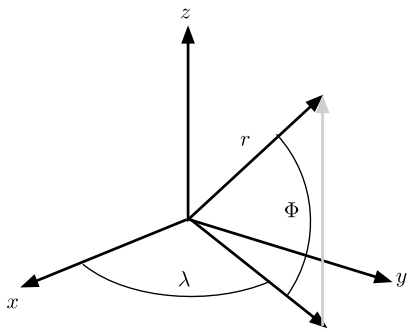


Figure I.5 Coordinates for a nonhomogeneous planet.

If we define i, j, k as unit vectors along the planetary x-, y-, and z-axes, the gravitational acceleration can be found as follows

$$\frac{\partial V}{\partial r} = \sum_{n=2}^{\infty}\sum_{m=0}^{n}\frac{\partial V_{n,m}}{\partial r} \tag{I.49}$$

and

$$\frac{\partial V_{n,m}}{\partial r} = \frac{\mu}{r^2}\left(\frac{a}{r}\right)^n \left[-r(vH_{n,m} + B_{n,m}) + iD_{n,m} - jE_{n,m} + kH_{n,m}\right] \tag{I.50}$$

$$B_{n,m} = (C_{n,m}\hat{C}_m + S_{n,m}\hat{S}_m)(n+m+1)P_n^m \tag{I.51}$$

$$E_{n,m} = -m(C_{n,m}\hat{S}_{m-1} - S_{n,m}\hat{C}_{m-1})$$

$$D_{n,m} = m(C_{n,m}\hat{C}_{m-1} + S_{n,m}\hat{S}_{m-1})$$

$$H_{n,m} = (C_{n,m}\hat{C}_m + S_{n,m}\hat{S}_m)P_n^{m+1}$$

$$\hat{C}_m = \hat{C}_1\hat{C}_{m-1} - \hat{S}_1\hat{S}_{m-1}$$

$$\hat{S}_m = \hat{S}_1\hat{C}_{m-1} + \hat{C}_1\hat{S}_{m-1}$$

The starting conditions are

$$\begin{aligned} \hat{C}_0 &= 1 \quad \hat{S}_0 = 0 \\ \hat{C}_1 &= \tfrac{x}{r} \quad \hat{S}_1 = \tfrac{y}{r} \end{aligned} \tag{I.52}$$

The derivatives of the Legendre polynomials are

$$P_n^0 = \frac{1}{n}\left[(2n-1)vP_{n-1}^0 - (n-1)P_{n-2}^0\right] \tag{I.53}$$

$$P_n^m = P_{n-2}^m + (2n-1)P_{n-1}^{m-1} \tag{I.54}$$

$$P_0^0 = 1$$

$$P_1^0 = \frac{z}{r} \equiv v$$

$$P_0^1 = 0$$

$$P_1^1 = 1$$

This kind of harmonic expansion is the standard method for modeling a planet's gravitational field. Harmonic models are available for the Earth, Moon, Mars, and other planets. A harmonic model is convenient because it gives an idea of how much influence higher-order terms have on the overall force.

An alternative approach is to use (I.47) and model the potential using a set of point masses. A combination of point masses and a spherical harmonic model can also be employed. This has been used for lunar-gravity models where concentrations of mass (mass cons) are modeled by point masses and the overall figure of the Moon is modeled using a harmonic expansion.

Sometimes the harmonics with $m = 0$ are given as J_n starting with J_2 ($J_1 = 0$). The largest term for the Earth is J_2, which is -1.0826×10^{-3}, due to the oblateness (or flatness of the poles compared to the equator) of the Earth. J_3 is 2.5326×10^{-6}. J_n is the same as $C_{n,0}$.

I.6. Linearized orbit equations

The central-body equations in cylindrical coordinates are

$$\ddot{r} - r\dot{\theta}^2 + \frac{\mu}{r^2} = a_x \cos\theta + a_y \sin\theta \tag{I.55}$$

$$r\ddot{\theta} + 2\dot{r}\dot{\theta} = a_y \cos\theta - a_x \sin\theta \tag{I.56}$$

$$\ddot{z} + \frac{\mu z}{r^3} = a_z$$

$$r^2 = x^2 + y^2 + z^2$$

These equations are still nonlinear. The linearized equations for a given R and constant orbit rate ω are

$$\ddot{x} - 3\omega^2 x - 2\omega\dot{y} = a_x \tag{I.57}$$

$$\ddot{y} + 2\omega\dot{x} = a_y \tag{I.58}$$

$$\ddot{z} + \omega^2 z = a_z \tag{I.59}$$

where x is radial deviation from the nominal R, y is $R\theta$ where θ is the orbit angle to the reference and R is the nominal orbit radius, and z is normal to the orbit. The disturbances have been redefined. Define $\sigma = \omega t$. The equations become

$$\ddot{x} - 3x - 2\dot{y} = \frac{a_x}{\omega^2} \tag{I.60}$$

$$\ddot{y} + 2\dot{x} = a_y \tag{I.61}$$

$$\ddot{z} + z = a_z \tag{I.62}$$

where the dot is now with respect to σ. These new equations are much better suited for numerical analysis than the previous set. This is because the resulting elements in the state-transition matrix are all the same order of magnitude. When the elements vary greatly in magnitude the smaller ones tend to get lost in many numerical routines. An interesting problem is that of a solar sail in orbit around the Sun with the sail normal along the radius vector. This results in a purely radial acceleration. The equations become

$$\ddot{x} - 3x - 2\dot{y} = \frac{a_x}{\omega^2} \tag{I.63}$$

$$\ddot{y} + 2\dot{x} = 0$$

The easiest way to solve this problem is with Laplace transforms,

$$\begin{bmatrix} s^2 - 3 & -2s \\ 2s & s^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{a_x}{s\omega^2} \\ 0 \end{bmatrix} \tag{I.64}$$

or

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{\begin{bmatrix} s^2 & 2s \\ -2s & s^2-3 \end{bmatrix}}{s^2(s^2+1)} \begin{bmatrix} \frac{a_x}{s\omega^2} \\ 0 \end{bmatrix} \tag{I.65}$$

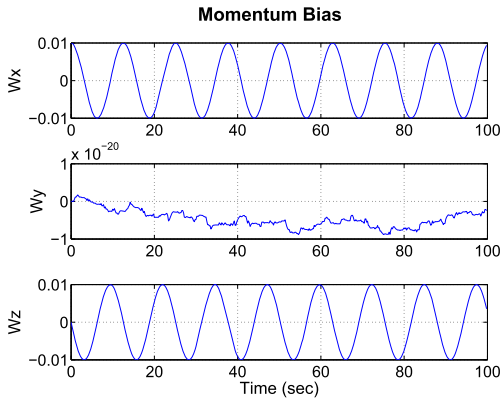
The solution is

$$\begin{aligned} x &= a_x(1-\cos\omega t) \\ y &= -2a_x(\omega t - \sin\omega t) \end{aligned} \tag{I.66}$$

A radial force causes an oscillation in the radial direction, but also causes the spacecraft to move in the circumferential direction relative to the initial position. Thus a radial force can be used to change the phase of the spacecraft.

The response of the inplane components of a linearized orbit is shown in Example I.2. The orbit rate is set to 1.

```
[a,b]   = LinOrb( [], 1 );
a       = a([1 2 4 5],[1 2 4 5]);
b       = b([1 2 4 5],[1,2] );
c       = eye(2,4);
d       = eye(2,2);
[m,p,w] = FResp(a,b,c,d);
yL      = {'|1,1|', '|1,2|', '|2,1|', '|2,2|'};
Plot2D(w,20*log10(m),'\omega (rad/s)', yL,...
   'Orbit Frequency response', 'xlog')
```



Example I.2: Linearized orbit-frequency response.

The z component is a simple oscillator. The crosschannels (i.e., disturbance in 2 to 1 position) have only zeros, no poles. The diagonal channels have pole zero pairs. All of the poles are at orbit rate. A disturbance in the radial direction will cause a steady offset in radius. However, a step radial disturbance will cause an infinite steady response in tangential position and a step tangential disturbance will cause an infinite steady response in body axes.

APPENDIX J

Optics

J.1. Optical sensors

Many attitude sensors are optical. Spacecraft optical sensors work in the infrared and visible bands. This appendix describes optical systems including lenses and telescopes. It also provides an introduction to geometric optics.

J.1.1 Optical nomenclature

Table J.1 gives optical concepts relevant to spacecraft sensing.

J.1.2 Telescope types

An imaging optical sensor consists of a telescope and an imaging chip. Fig. J.1 shows a hierarchy of telescope types. At the primary level, telescopes can be classified based on whether they employ reflective (mirrors) elements or refractive elements (lenses), or a combination of both. The greatest variety of telescopes is found among the reflectors. Fig. J.2 shows the optical geometry of the telescopes shown in Fig. J.1.

1. Refracting telescopes use a combination of an objective lens and an eyepiece lens. The objective lens defines the aperture of the telescope. The eyepiece lens causes the rays to converge. Refractors have relatively high spherical aberration and chromatic aberration, which is worse with short focal lengths. They also have light losses through the objective lens.
2. The Schmidt–Cassegrain has a corrector plate as the first optical element. This corrects for the spherical aberration of the primary mirror. It has light losses due to the obstruction.
3. The Maksutov–Cassegrain uses a meniscus corrector shell that is a section of a sphere. It has a spherical secondary mirror. The Maksutov has better resolution than the Schmidt. It suffers some light losses through the first optical element.
4. The Ritchey–Chretien is a Cassegrain reflector with two hyperbolic mirrors instead of a parabolic mirror. It can be free of coma and spherical aberration.
5. The Dali–Kirkham uses a concave elliptical primary mirror and a convex spherical secondary mirror. It does not correct for offaxis coma or field curvature.
6. The Schiefspiegler uses a tilted mirror to avoid the secondary mirror. This eliminates diffraction patterns but leads to an increase in coma and astigmatism.
7. The Gregorian employs a concave secondary mirror.

Table J.1 Nomenclature.

Term	Description
aperture stop	is a ring at the aperture that limits the size of the aperture. This is seen in cameras. The radius of the first lens serves as an aperture stop. The aperture stop controls the number of rays reaching the imaging plane but does not block all rays from a point on the imaged object. This is always behind the first lens.
achromat	a lens that reduces chromatic aberration. Often a doublet with lenses of different glasses.
apochromat	a lens that has less chromatic aberration than an achromat lens.
chief ray	is a ray that starts at the edge of an object and passes through the center of the aperture.
chromatic aberration	is the phenomenon of different wavelengths of light focusing at different points since the index of refraction is wavelength dependent.
coma	is an aberration in which parallel rays (for example from a star), at an angle to the optical axis of a lens focus at different points even though they would focus at the same point were the rays parallel to the optical axis. This means that stars on the edge of the field of view will be v-shaped with a comet-like tail.
field stop	is a ring at the imaging plane that determines the field of view of the instrument. The boundary of the CMOS chip acts as a field stop. The field stop obstructs certain rays completely.
meridional ray	is a ray that is confined to the yz-plane in an optical system in which z is along the optical axis and y is perpendicular to the axis.
paraxial ray	is a ray that makes a small angle to the axis system so that the angle of incidence $\theta \approx \sin\theta$.
sagittal ray	is a skew ray that enters the entrance pupil at $y = 0$.
skew ray	is a ray that starts in the yz-plane but enters the entrance pupil with a nonzero x coordinate.
vignetting	is the reduction of an image's brightness near the periphery. It can be caused by external objects, the physical dimensions of multiple lenses, and by the angle dependence of the image-sensor elements. The use of microlenses can reduce the pixel problem. Optical vignetting can be cured by reducing the aperture diameter.

8. Nasmyth reflectors use the third mirror to reflect light to the side. This adds flexibility. A prism can be used so that multiple targets are available. However, the prism adds to light losses.

Note that when the image is only to be viewed through an imaging chip no eyepiece is required. The imager is placed at the focal plane of the telescope. The simplest telescope would be a single objective lens and an imaging chip.

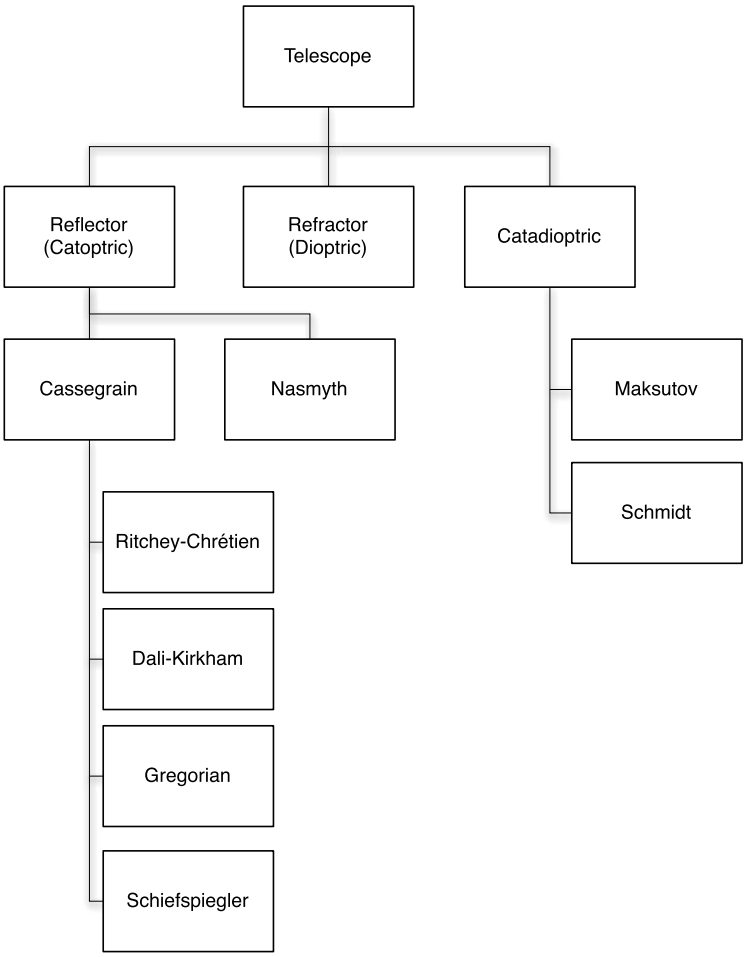


Figure J.1 Telescope taxonomy shows the variety of telescopes that could be used as part of a spacecraft optical system.

J.1.3 Geometry of imaging

The geometry of imaging is shown in Fig. J.3. This shows the relationship between the point on the image and the point on the focal plane.

In this diagram the subscript o is for object and i is for image. The blue lines are rays from the object point and all converge on the same point on the image plane. The position of the point r_o on the image plane is

$$|r_i| = r_o \left(\frac{f}{s_o - f} \right) \tag{J.1}$$

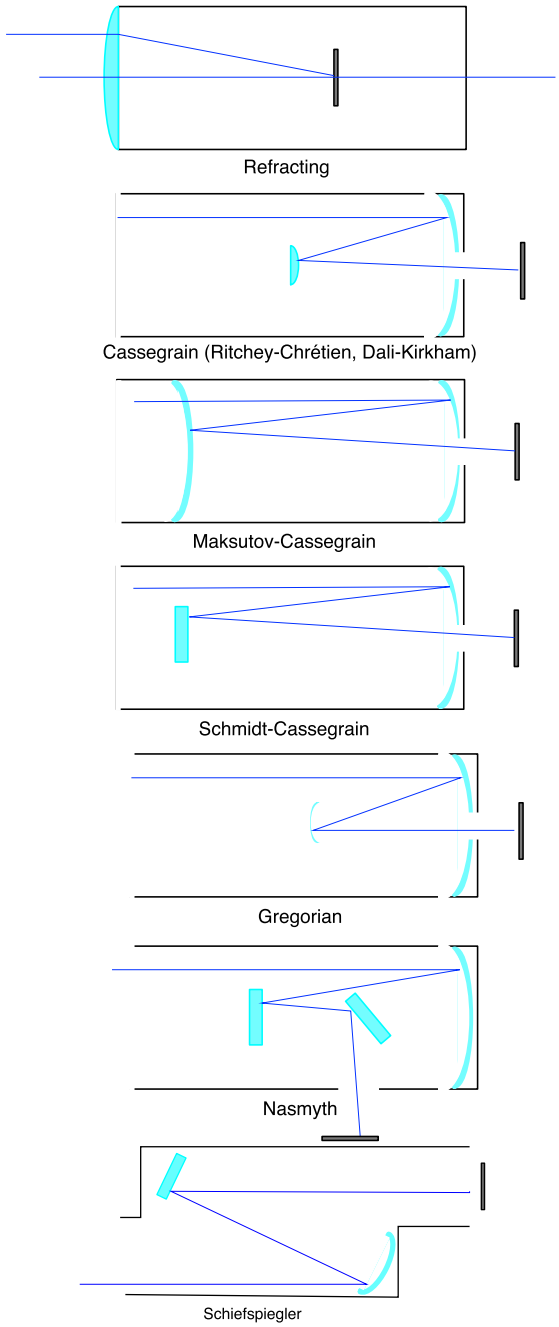


Figure J.2 Telescope optical geometry. The gray rectangle is the imaging chip.

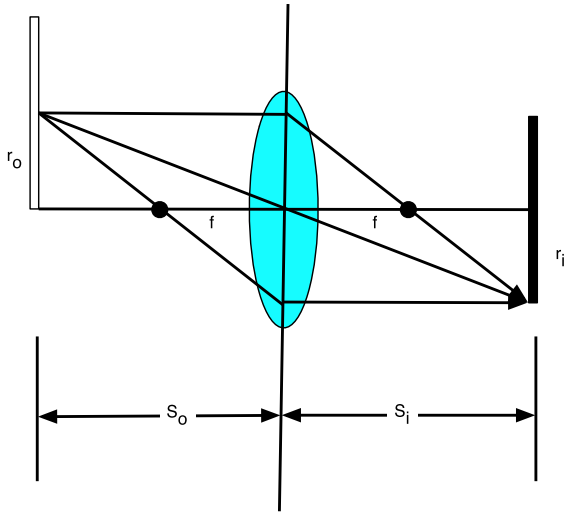


Figure J.3 Geometry of imaging for a simple lens.

where f is the focal length of the camera and s_o is the distance of the object from the camera lens. If the lens is circularly symmetric, then r_i is the distance of the point from the imaging axis. This formula assumes thin lens. The lens formula is

$$\frac{1}{f} = \frac{1}{s_o} + \frac{1}{s_i} \tag{J.2}$$

This is identical for a mirror in which for a spherical mirror $f = -R/2$, where R is the mirror radius. The reflecting geometry is shown in Fig. J.4. If the target is many focal lengths away we find that

$$s_i = f \tag{J.3}$$

Fig. J.4 shows the definition of the focal point for a mirror.

At a more fundamental level, optical-system performance is based on Snell's law of refraction [1] shown in Fig. J.5. We arrange lenses so that incoming rays from objects of interest arrive at the desired spot on the imaging chip.

$$\frac{\sin\theta_i}{\sin\theta_t} = \frac{n_2}{n_1} \tag{J.4}$$

and Alhazen's law of reflection

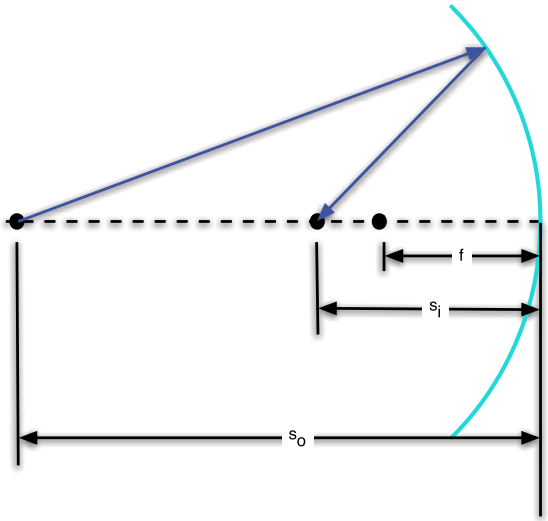


Figure J.4 Mirror optical geometry.

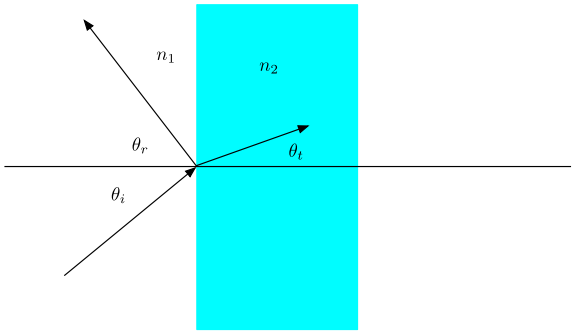


Figure J.5 Snell's law.

$$\theta_i = \theta_2 \tag{J.5}$$

where n_k is the index of refraction. The index of refraction is a function of the wavelength of the incoming light. Different wavelengths of light will not focus onto the same spot when a single lens is employed. Different lenses of different refractive indices are used to compensate for this phenomena.

Lenses are either positive or negative [2], which defines the sign of their focal length. Either surface can be concave, convex, or flat, see Fig. J.6.

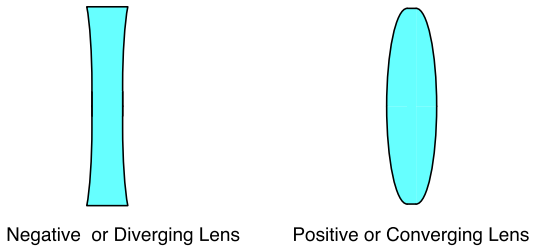


Figure J.6 A negative lens is on the left and a positive lens is on the right.

J.1.4 Telescope performance

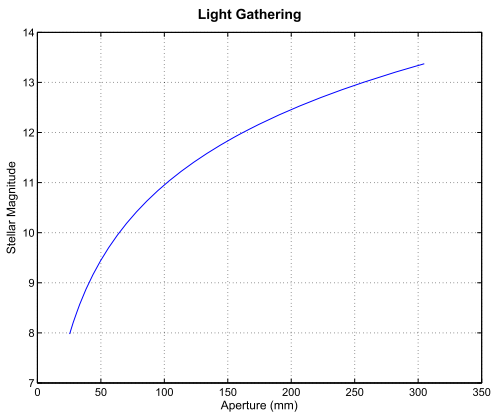
J.1.4.1 Geometric

The angular resolution of a telescope is limited by the aperture. The Rayleigh criterion for the resolution limit is given by

$$\alpha_D = 1.22\frac{\lambda}{D} \tag{J.6}$$

where D is the aperture diameter and λ is the wavelength of the light.

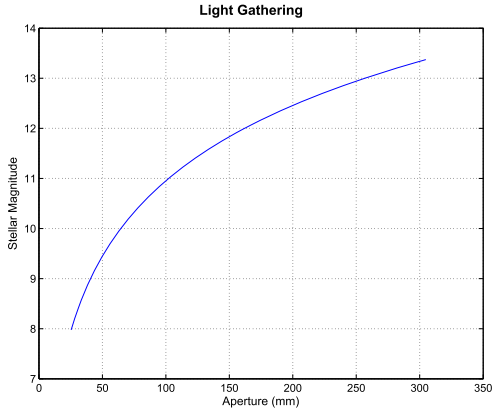
```
lambda = linspace(400,700)*1e-9;
f = 100;% Focal length
a = 50; % Aperture
F = f/a; % FStop
d  = AiryDisk( lambda, F );

Plot2D( lambda*1e9, d*1e6, '\lambda (nm)', '
      Diameter (\mu m)', 'Airy Disk' );
PrintFig(1,1,1,'DiffractionLimitedResolution')
```



Example J.1: Diffraction-limited resolution.

This limit is based on the first zero-crossing of the Airy pattern for a point source (such as a star). This is shown in Example J.2. It shows the pattern with and without a central obstruction. Most reflecting telescopes have a central obstruction. It is possible to separate points that are closer than this value if the individual pixels are close enough together and the individual pixels have sufficient dynamic range. Such techniques are sometimes called superresolution techniques.

```
d      = 100;% Aperture
f      = 80;% Focal length
lambda = 650*1e-6;
eps    = 0.5;% Central obstruction
[i1, pixels] = AiryResolution( d, f, 3.5e-3,
       lambda );
[i2, pixels] = AiryResolution( d, f, 3.5e-3,
       lambda, eps );
Plot2D(pixels,[i1;i2],'Pixels','Amplitude','Airy␣
       Disk')
legend('Unobstructed',sprintf('%2.0f%%␣obstructed
       ',eps*100))
PrintFig(1,1,1,'AiryPattern')
```



Example J.2: Airy pattern for a point source.

The spatial resolution is

$$r_D = f\alpha_D \tag{J.7}$$

The image size of an object of angular size θ is

$$w = f\theta \tag{J.8}$$

If the sensor has p pixels per unit length then the angular resolution of each pixel is

$$\theta_p = f\theta/p \tag{J.9}$$

For a star image the star will be a point and it will be desirable to defocus the point so that it is spread over several pixels. The light-gathering ability of a telescope is proportional to the square of the aperture

$$L \propto \pi \frac{D^2}{4} \tag{J.10}$$

Losses include losses through lenses, the surfaces of mirrors and those due to obstructions (such as in a reflecting telescope) [3]. The best coatings on mirrors reduce losses to 2%. Losses in lenses are 1% per inch for inglass absorption. Light loss due to a central obstruction is

$$l_o = \frac{13a}{D} \tag{J.11}$$

in percent where a is the diameter of the obstruction.

Both light gathering and resolution improve with larger apertures. However, larger apertures result in more mass and volume. The f-ratio or f-number of a telescope is

$$F = \frac{f}{D} \tag{J.12}$$

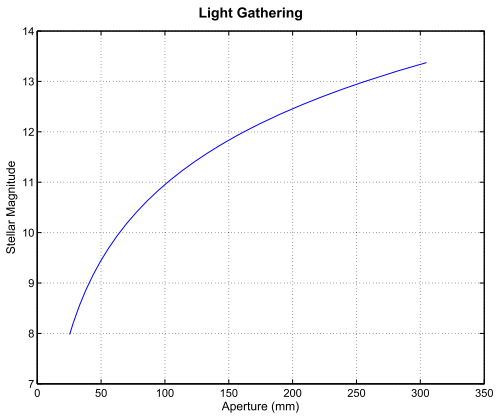
The greater the f-number the less light reaches a given area on the focal plane.

The stellar magnitude that can be seen by an ideal telescope is

$$m = 2.7 + 5 \log D \tag{J.13}$$

where D is the aperture in mm. Example J.3 shows the light-gathering capability of a telescope.

```
1 LightGathering;
2 PrintFig(1,1,1,'LightGathering');
```



Example J.3: Light-gathering capability of a telescope.

Example J.4 shows the visible stars as a function of telescope aperture using the Hipparcos star catalog. The sensor efficiency (the product of the fill factor and the sensor quantum efficiency) is 0.2. Three lenses are assumed, two for the objective and one for the field corrector each with a light-transfer efficiency of 0.99. A 160-mm aperture is sufficient to see the entire catalog.

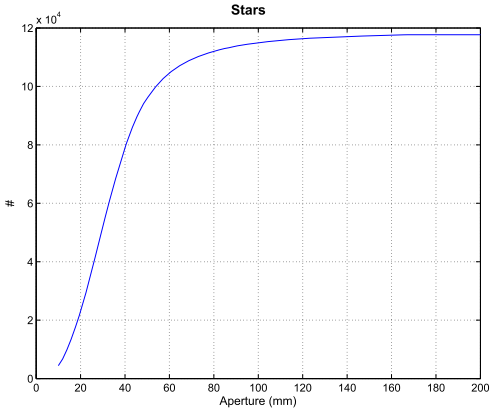
J.1.4.2 Errors

The first-order monochromatic errors can be decomposed into the following five types: spherical aberration, coma, astigmatism, field curvature, and distortion.

Spherical aberration is the difference in focal length between axial rays and rays along the periphery of aperture. These errors are proportional to the square of the aperture.

Coma is a geometric property in which offaxis point sources do not all come together at the same point. In an ideal telescope all of the light gathered by the aperture

```
nLenses   = 3;
nSurfaces = 2*nLenses;
eff       = 0.2*0.99^nSurfaces;
c         = LoadCatalog('Hipparcos',12);
n         = 100;
nStars    = zeros(1,n);
d         = linspace(10,200,n);
for k = 1:n
    m = LightGathering( d(k), eff);
    nStars(k) = length(find( c.vM< m));
end
Plot2D( d, nStars, 'Aperture (mm)', '#', 'Stars')
      ;
PrintFig(1,1,1,'NStars');
```



Example J.4: Visible stars for an ideal aperture.

from a source at infinity should appear at the same point. Schmidt, Maksutov, and other telescopes are designed to reduce coma.

Monochromatic or third-order astigmatism is a geometric effect even in symmetric optical systems. It is sometimes called third-order astigmatism. The larger the angle between the ray from the viewed object and the axis of the system the larger the astigmatism.

Field curvature means that the image really lies on the surface of a sphere, not a flate plate. This causes offaxis errors when an imaging chip is used.

Barrel distortion is an effect in which magnification decreases with distance from the optical axis. Pincushion distortion is the opposite.

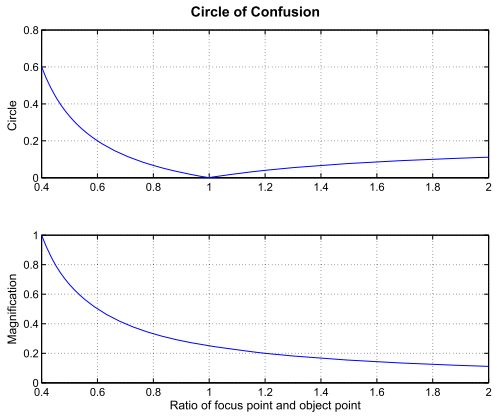
There are also the chromatic aberrations decomposed into longitudinal and transverse. Additional errors are lens flare and diffraction.

Lens flare is light scattered in a telescope by internal reflection and scattering when a bright source, such as the Sun, is in the field-of-view. Lens flare often occurs in lenses with multiple elements such as zoom lenses. Lens flare can also occur when the bright object is not in the field-of-view in which case it appears as a haze that reduces contrast. Internal diffraction on the image sensor can cause small rainbows in the image.

Diffraction spikes are an artifact caused by light diffracting around the support vanes of a secondary mirror in a reflecting telescope. Baffles and other objects can also cause diffraction errors. These patterns can cause errors in centroiding. These errors can be minimized by employing a centroiding algorithm that uses the point-spread function for a point source that will then ignore the spikes. This is also reduced if the image is defocused since only sharp objects have distinctive diffraction artifacts.

The circle of confusion is the spreading of an ideal point from the light from a lens, which is focusing a point source, not coming to a perfect point. This is also known as the blur circle. This is illustrated in Example J.5.

```
s2 = 10;% Object location
s1 = linspace(4,20);% Object point
a  = 1;% Aperture
f  = 2;% Focal length
CircleOfConfusion( s1, s2, f, a );
PrintFig(1,1,1,'CircleOfConfusion');
```



Example J.5: Circle of confusion.

J.1.5 Pinhole camera

The lowest-order approximation to the optical system is a pinhole camera. The pinhole camera is shown in Fig. J.7.

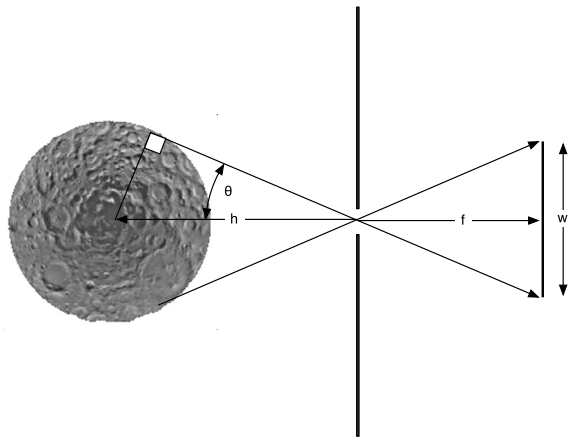


Figure J.7 Pinhole camera.

A point $P(X, Y, Z)$ is mapped to the imaging plane by the relationships

$$u = \frac{fX}{Z} \tag{J.14}$$

$$v = \frac{fY}{Z} \tag{J.15}$$

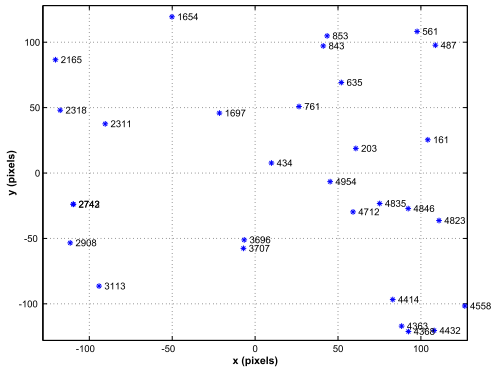
where u and v are coordinates in the focal plane, f is the focal length, and Z is the distance from the pinhole to the point along the axis normal to the focal plane. This assumes that the Z-axis of the coordinate frame X, Y, Z is aligned with the boresight of the camera.

The angle that is seen by the imaging chip is

$$\theta = \tan^{-1}\left(\frac{w}{2f}\right) \tag{J.16}$$

where f is the focal length. The shorter the focal length, the larger the image. The pinhole camera does not have any depth of field, but that is unimportant for many far-field imaging problems. The field-of-view of a pinhole camera is limited only by the sensing element. An example of a pinhole image from the Hipparcos catalog (showing only stars with a visual magnitude >6) is given in Example J.6.

```
1 PinholeCamera
2 PrintFig(1,1,1,'PinholeCameraDemo');
```



Example J.6: Pinhole-camera star image.

J.2. Radiometry

J.2.1 Mathematical basis for position and attitude determination using a camera

The elements of a vision-based system are shown in Fig. J.8. The external sources of illumination for a satellite are nearby planets and the Sun. Since we are concerned about imaging we must consider the spectrum of the incoming radiation from each source. This plot gives the illumination spectrum. Radiation from the Moon and Earth is in the infrared part of the spectrum so will not be seen by cameras using conventional silicon imaging arrays. The Earth's black-body temperature is 295 K, the Moon's is 104 K, and the Sun's is 5760 K.

This also means that the illumination source for the spacecraft can be assumed to be a point, not a distributed source as radiation from a nearby planet would be modeled.

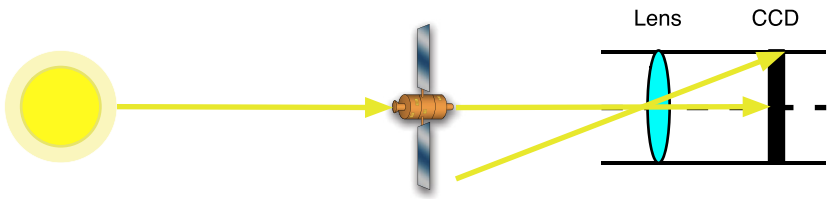
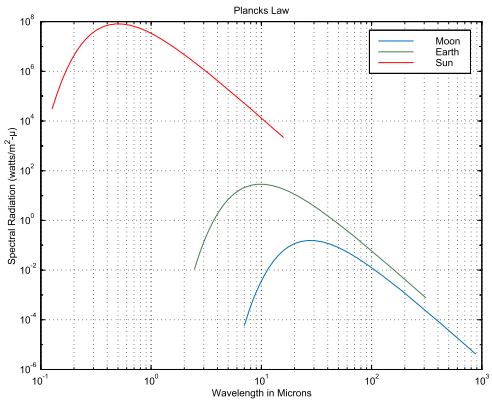


Figure J.8 Vision system.

```
1 tMoon  = 104;
2 tEarth = 295;
3 tSun   = 5760;
4
5 PlanckL([tMoon;tEarth;tSun]);
6 legend('Moon','Earth','Sun')
7 PrintFig(1,1,1,'PlanckEx')
```



Example J.7: Earth, Moon, and Sun spectra.

J.2.2 Basic radiometry

Fig. J.9 illustrates the basic radiometric concepts. The simplest illumination model is the Lambertian surface-reflection model

$$L = \rho I^T n \tag{J.17}$$

where n is the normal to the surface, I is the vector to the illumination source, and $0 \leq \rho \leq 1$ is the surface's albedo.

According to the Lambertian model the Lambertian surface reflects light in a given direction d proportionally to the cosine of the angle between d and n. However, since the surface area seen from direction d is inversely proportionally to the angle between d and n, the angles cancel and d does not explicitly appear in the equation. In Fig. J.9 we have shown one source (the Sun); however, reflections from other surfaces will also be sources, thus complicating the illumination model. The energy received from the object is

$$E(p) = L(P)\frac{\pi}{4}\left(\frac{a}{f}\right)^2 \cos^4 \alpha \tag{J.18}$$

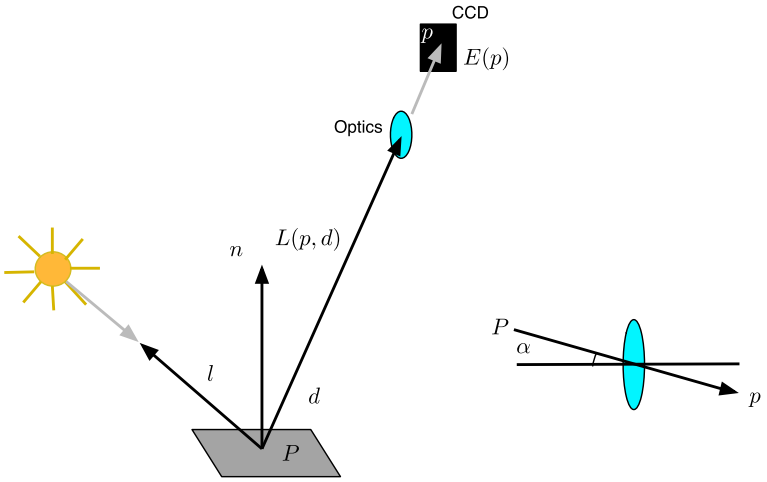


Figure J.9 Basic radiometry.

where a is the diameter of the aperture, f is the focal length, α is the angle between the incoming light and the optical axis, and f/a is the F-number of the optical system. For most optical systems α is small so the last factor is nearly 1. $E(p)$ is in units of W/m^2 while $L(p)$ is in units of W/m^2 sr.

J.2.3 Radiosity

As mentioned above, each surface will be illuminated not just by the Sun, but also by radiation emitted from other surfaces. The Lambertian surface-reflection model assumes that the surfaces do not interact with each other and are not a source of illumination. A more general model is a radiosity model. Radiosity models are similar to models used for radiative thermal analysis of spacecraft.

The radiosity method generates a "photorealistic" model of the spacecraft that can be used as part of the vision system. There are two major parts to radiosity-based imaging:

1. Generation of form factors;
2. Solution of the radiosity equation.

The form factor between two planar surfaces is

$$F_{E_i \to E_j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_i dA_j \tag{J.19}$$

Note that this requires numerical integration over the finite area. In practice, if the surfaces are separated by more than five times their area, the integral can be replaced by a constant over the surfaces. The radiant exitance is then found from the radiosity

equation

$$M_o = \begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & 1-\rho_n F_{nn} \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \cdots \\ M_n \end{bmatrix} \tag{J.20}$$

where

$$M = \pi L \tag{J.21}$$

This matrix equation has one row and one column for each patch on the object. The column vector on the right is needed to compute the light reaching the camera. The vector on the left-hand side would include external sources, in this case the Sun. This equation is solved by a method called progressive refinement, which provides approximate solutions that improve with each iteration.

J.2.4 Radiometric sources

Radiometric sources for spacecraft optical include other spacecraft, the Sun, planets, and other solar-system bodies and stars. The stellar magnitude that can be seen by a telescope is shown in Example J.4.

Solar-system bodies will be illuminated by the Sun. Table J.2 lists the albedo and radiation reflected from each body at their mean distance from the Sun. The bond albedo is the ratio of the total radiation reflected to the incident radiation from the Sun. The geometric albedo is the fraction of the radiation relative to that from a flat Lambertian surface, which is an ideal reflector at all wavelengths.

The Sun's radiation in W/m^2 is

$$p = \frac{1367}{r^2} \tag{J.22}$$

where r is distance from the Sun in astronomical units. 1367 is the flux of the Sun at the mean orbital radius of the Earth.

The albedo flux from any planet is

$$p = \frac{p_1 R^2}{r^2} \tag{J.23}$$

where r is the distance to the satellite, R is the planet radius, and p_1 is the albedo flux at the surface of the planet.

Table J.2 Reflected light from planets and moons.

Planet	Geometric Albedo	Bond Albedo	Albedo Flux based on the Bond Albedos (W/m^2)
Mercury	0.138	0.056	510.873
Venus	0.84	0.720	1881.160
Earth	0.367	0.390	533.130
Mars	0.15	0.160	94.213
Jupiter		0.700	35.343
Saturn		0.750	11.272
Uranus		0.900	3.340
Neptune		0.820	1.240
Pluto	0.44–0.61	0.145	0.127
Europa		0.670	33.828
Triton		0.760	1.149
Earth's moon	0.113	0.120	164.040
Titan		0.210	3.156

J.2.5 Noise and performance factors

J.2.5.1 Fill factor

Fill factor is the percentage of the sensor that is active. In a CCD this is nearly 100%. In APS or CMOS sensors this can be considerably smaller due to the additional support circuits.

J.2.5.2 Illumination side

Chips can be back or front illuminated. Back illumination means the illumination is on the side opposite that of the circuit elements. This eliminates the blockage due to the traces increasing the fill factor to 100%. Most modern sensors use back-illuminated chips.

J.2.6 Dynamic range

The range of photon fluxes that can be accommodated without saturation. The dynamic range is determined by the size of the pixel and the number of electrons it can hold.

J.2.7 Blooming

Blooming is when charge from one detector overflows into another. In a CCD, if the cell saturates it will overflow into other cells. Antiblooming devices can be added to CCD chips at the expense of fill factor.

J.2.8 Quantum efficiency

Quantum efficiency is the percentage of photons hitting a detector that product a hole–electron pair, that is an output. CCDs have quantum efficiencies of over 90% at some wavelengths making them well suited for low-light (or deep-sky) imaging. CMOS chips have quantum efficiencies that are around 30%.

J.2.8.1 Dark current

This is the constant output of a detector when it is not illuminated. The dark current will appear in every exposure. Since it is constant it can be measured and subtracted. This is best done on-orbit. CCD chips have very low dark current, as low as 1 electron/second at −30°C. CMOS cells are considerably higher. Part of this is due to the manufacturing of CMOS sensors in standard CMOS fabrication plants. CCD sensors that are used for imaging are fabricated on fabs specialized for imaging chips. As CMOS use increases dark current will improve. With all sensors dark current can be improved by cooling. For spacecraft applications and many terrestrial applications, thermoelectric coolers are used for imaging-chip cooling. These are solid-state devices and can be integrated into the other sensor electronics.

J.2.8.2 Fixed-pattern noise

Fixed-pattern noise is noise over the entire array that occurs in the same pattern on every exposure.

J.2.8.3 Readout noise

This is the noise added to a pixel output on reading it out.

J.2.8.4 Radiation hardness

Radiation hardness is the ability of the chip to withstand long-term and prompt radiation dosages. Prompt dosages can cause single-event upsets, in which a switch is set incorrectly, for example a readout switch. A greater problem is latchup that requires power to be cycled to restore the functioning of the chip. Long-term damage can increase the noise and reduce the quantum efficiency.

J.2.9 Imaging-chip theory

J.2.9.1 Dark current

The rate of dark-current creation is [4]

$$S = AT^{3/2}e^{-\frac{V_g q}{2kT}} \tag{J.24}$$

where V_g is the gap voltage, T is the temperature in K, q is the electron charge (1.6×10^{-19} C), k is Boltzmann's constant (1.38×10^{-23} J/K), and A is 3.36×10^6 when S is

in volts. For a typical detector V_g is 1.36 V. To convert this to the number of electrons

$$e = \frac{St}{Cq} \tag{J.25}$$

where C is the imaging-chip capacitance since

$$S = C\frac{dV}{dt} \tag{J.26}$$

If the dark current is known it can be subtracted. However, this requires measurement of the current in the dark for each pixel. This is known as a thermal map. This equation shows it is advantageous to cool the chip, which can be done with a thermoelectric cooler. There are limits on how cold the chip can be cooled since the spectral sensitivity changes. Charges of thermal origin are created in the imaging elements and transfer elements.

J.2.9.2 Cosmic rays

A cosmic-ray impact will cause the generation of charge in a pixel. Cosmic rays are indistinguishable from stars. While not an issue with planetary imaging this can cause the sensor to misidentify stars. The simplest method is to take two images and compare them. The cosmic-ray image will only appear in one of the frames. The star identification software will generally also ignore "false stars", particularly when the sensor is tracking known stars but the safest method is the two-frame method.

J.2.9.3 Thermal noise

Thermal noise is a function of the level of dark current. If N_t is the number of thermal charges created by the dark current then

$$\sigma_o = \sqrt{N_t} \tag{J.27}$$

J.2.9.4 Transfer efficiency

Every time charge is transferred noise is created. The noise created is

$$\sigma_e = \sqrt{2\epsilon n N_s} \tag{J.28}$$

where ϵ is the transfer efficiency, n is the number of transfers, and N_s is the charge transferred that includes thermal electrons. For an APS, CID, or CMOS sensor n = 1. For a CCD n = the number of cells between the cell of interest and the readout.

J.2.9.5 Reset noise

At the output of a CCD register the floating readout diode is brought to a reference voltage before a packet is read out. This creates "reset noise", which is

$$\sigma_r = \frac{1}{q}\sqrt{kTC} \tag{J.29}$$

where C is the capacitance of the output diode. This noise can be eliminated by double-correlated sampling.

J.2.9.6 Photon noise

Photon noise is a result of photons hitting the chip and is

$$\sigma_s = \sqrt{QN_p} \tag{J.30}$$

where Q is the quantum efficiency and N_p is the number of photons hitting the chip.

J.2.9.7 Quantization noise

Quantization noise is

$$\sigma_q = \frac{N}{2^n\sqrt{12}} \tag{J.31}$$

where N is the maximum number of electrons per at the imaging-chip readout and n is the number of bits.

J.2.9.8 Total noise

The total noise is found by summing the squares of the noise sources

$$\sigma_t = \sqrt{\sigma_o^2 + \sigma_e^2 + \sigma_r^2 + \sigma_s^2 + \sigma_q^2} \tag{J.32}$$

J.2.9.9 Blooming

When a photoelement is subject to strong illumination it can overflow and spill charges into adjacent photoelements if there is a direct connection between photoelements as there is in a CCD. CMOS, APS, and CID sensors do not have such a path and are immune to blooming. Another effect is that the charge cannot be emptied in one transfer leaving a "trail" of charges in that element. These leftover charges must be removed during successive readouts but it means that each readout will be contaminated. This is not to say that there is no crosstalk between sensing elements. All APS and CMOS chips specify crosstalk as a function of element separation.

CCDs can employ antiblooming devices to drain the overflow. Some of the excess charge will still remain. In addition, these devices take up space on the chip reducing its overall light-gathering capability. Typically, such devices reduce the sensitivity by 10%.

J.2.9.10 Linearity

The response curve of a photoelement is not entirely linear. At high levels the elements saturate. At low levels, the response is not proportional to the input. In most photoelements the nonlinearity at low signal levels is negligible. As an example the Cypress STAR 1000 sensor has a full-well capacity of 135 000 electrons but is linear to within 1% only with 95 000 electrons.

J.2.9.11 Amplifier noise

CMOS image sensors suffer from higher noise than CCDs due to the additional pixel and column amplifier transistor noise.

J.2.10 Data reduction

The following steps are taken to reduce data for an imaging chip [5]. Not all are applicable to CMOS chips. Several of the steps require calibration with either uniform sources or no illumination.

1. Bias subtraction - this is the removal of a fixed voltage to all elements. This is done by the subtraction of a bias frame.
2. Dark-current correction - this is the removal of the dark current. This is done by measuring the elements at the same temperature at which the exposure is taken and subtracting the dark frame. Hole-accumulation diodes (HAD) patented by Sony, can reduce the dark current at the source.
3. Flat-field correction - several images are taken under uniform illumination. These frames are averaged to produce the flat-field frame in which the frame contains the variations due to chip imperfections. The corrected frames (from the two steps above) are divided by this frame.
4. Cosmic-ray correction - this is done by taking several images of the same scene and removing artifacts that do not appear in all of the images.
5. Saturation trail - if a pixel is saturated the extra electrons will spill along the row. Any narrow feature along a row should be removed by postprocessing. This is a problem with CCD chips and does not appear in CMOS chips.
6. Diffraction features - support structure, baffles, and other objects can cause diffraction features. These patterns can cause errors in centroiding. These errors can be minimized by employing a centroiding algorithm that uses the point-spread function for a point source that will then ignore the spikes seen in the picture. This is also reduced if the image is defocused since only sharp objects have distinctive diffraction artifacts.

7. Bad column - in a CCD if a column is bad it will appear black and needs to be removed from the image. Any objects that intersect the bad column will be difficult to centroid. If a bad column is detected then the telescope should be pointed so that the objects of interest (for example an asteroid) do not appear in that column. This does not happen in CMOS chips.
8. Trap - in a CCD a damaged pixel may prevent reading of any pixels below that pixel. This will create artifacts in the image that must be avoided. This does not happen in CMOS chips.
9. Dust - dust on the lens will appear as constant dark areas. The flat-field correction will partially correct for these artifacts.
10. Photometric correction - Using objects of known brightness convert electrons produced by the detectors to flux.

References

[1] E. Hecht, Optics, 2nd edition, Addison-Wesley, 1990.
[2] W.J. Smith, Modern Lens Design, 2nd edition, McGraw-Hill, 2005.
[3] S. Vladimir, Amateur telescope optics, http://www.telescope-optics.net/index.htm, 2006.
[4] C. Bull, CCD Astronomy, Willman-Bell, Inc., 1991.
[5] O. Hainaut, Basic image processing, http://www.sc.eso.org/~ohainaut/ccd/CCD_proc.html, 1996.

APPENDIX K

Star-camera algorithms

K.1. Space story

I was working at GE Astro Space when I came up with the idea of the Polaris sensor, a bright-star sensor that would look North on a geosynchronous communications satellite. GE Pittsield had developed charge-injection device (CID) imagers that did not suffer from many of the problems inherent in charge-coupled device (CCD) imagers. I got IR&D money for the sensor and we began design work. A major problem turned out to be reflections from the solar wings that might blind the sensor due to reflections from the lens barrel. We came up with the idea of tilting it offaxis. Unfortunately, it was not a Polaris sensor. Eventually the program was canceled.

K.2. Introduction

Star cameras are a popular sensor for spacecraft-attitude determination. The idea is to measure the positions of stars in the imager frame and compare those with a star catalog. If the sensor measures at least three stars then 3-axis attitude determination can be done with one sensor. Often the sensor is coupled with a gyro. The star camera provides an initial attitude measurement and then corrects for gyro drift.

This chapter will discuss the process of going from an image of stars to locations of those stars in the field-of-view. More details on how they are then used for attitude determination is found in the estimation chapters.

K.3. Center-of-mass star centroiding

The first step is to get a coarse location of each star in the focal plane. The camera deliberately defocuses the stars so that they are spread over multiple pixels. This allows the algorithm to find a centroid with an accuracy that is not limited by the pixel size.

This section describes the initial "coarse" centroiding performed on an image. This centroiding is sufficiently accurate for star identification. This is a standard technique for centroiding stars in a star tracker. Fig. K.1 shows the steps required in the algorithm. First, the sky background must be modeled. This can be done with a Gaussian fit to the image histogram. The resulting background level b and standard deviation R allow for the pixelmap to be thresholded, and all pixels below the noise floor to be zeroed out. The remaining pixels comprise the imaged stars. In practice, the algorithm may be able to use a fixed threshold once the inspace background is modeled during on orbit

checkout. The threshold is based on the range of stellar magnitudes in the catalog that is used for star identification.

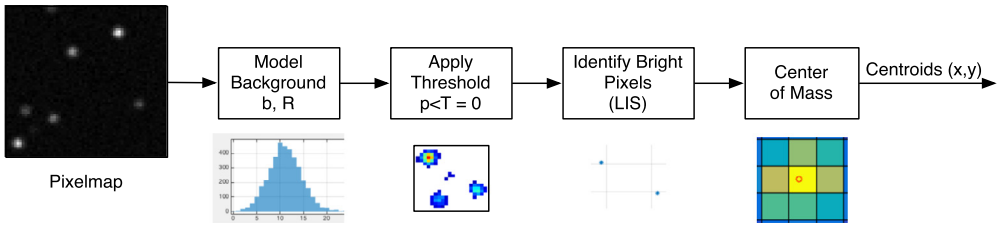


Figure K.1 Coarse centroiding steps.

A blobification routine can group the remaining pixels into blobs and find the brightest pixel in each. The reason it is called "blobification" is because the star images are not regular shapes. This is required in "lost-in-space" operation when the spacecraft has no estimate of its attitude. The routine eliminates blobs that are smaller than a specified minimum number of pixels. During tracking mode, when stars have already been identified and the spacecraft is rotating slowly, not all of the above steps may be required. Windows of the frame around the previously identified stars may be analyzed instead of the entire pixelmap and a local threshold may be applied.

K.3.1 Background noise

The background noise generates a Gaussian-like distribution of low pixel levels throughout the frame. This must be removed from the signal. One method is to fit a Gaussian to the histogram of the pixel counts, using the principal maximum of the histogram as the initial guess for the height of the curve [1]. Newton's method of nonlinear least squares usually gives a solution in two iterations. From the histogram, n_k are the discrete counts for each discretized intensity value I_k, see Fig. K.2. The Gaussian is expressed as,

$$f(I) = H \exp[-(I-b)^2/2\sigma^2] \tag{K.1}$$

The initial value for the mean background level b is the I_k with the maximum number of counts, i.e., the mode of the histogram. H is the height of the curve. When performing the least-squares fit it is best to scale the histogram counts n_k by the maximum value N. In this case the initial value for H is 1. σ can be initialized to the square root of b.

The least-squares δ for each iteration is

$$\delta = -(J^T J)^{-1} J^T f(I_k) \tag{K.2}$$

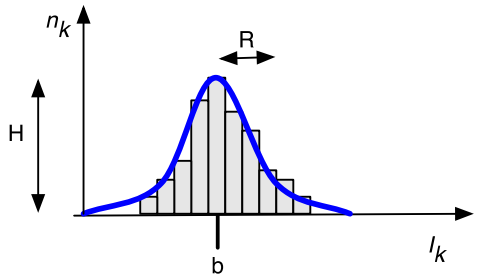


Figure K.2 Pixel histogram. *b* is the mean background level.

where the Jacobian matrix J of partial derivatives is

$$J_{ij} = \frac{\partial f_i(I)}{\partial x_j} \tag{K.3}$$

for the state elements $x = [b\, H\, \sigma]$ and an array of I_k around the initial value. In this case we can calculate the partial derivatives of the Gaussian distribution directly and avoid a numeric Jacobian,

$$\frac{\partial f}{\partial H} = \exp[-(I-b)^2/2\sigma^2] \tag{K.4}$$

$$\frac{\partial f}{\partial b} = -\frac{H}{\sigma^2}(I-b)\exp[-(I-b)^2/2\sigma^2] \tag{K.5}$$

$$\frac{\partial f}{\partial \sigma} = -\frac{H}{\sigma^3}(I-b)^2\exp[-(I-b)^2/2\sigma^2] \tag{K.6}$$

Two or three iterations are sufficient to calculate b and σ to several digits.

Once the sky-background level b is known, the pixel array can be thresholded as follows:

$$I(x,y) = I(x,y) - b \quad \text{if} \quad I(x,y) \geq T \tag{K.7}$$

$$I(x,y) = 0 \quad \text{if} \quad I(x,y) < T \tag{K.8}$$

The threshold level T must be selected on the basis of b and σ, for example at the 3σ level, this will be $T = b + 3\sigma$.

Fig. K.3 shows the effects of thresholding a simulated image. The image is first generated by integrating the Gaussian point-spread function for a set of sources, as shown on the left. Noise is then added using a Poisson process, center. A fit to the background of the image is performed and a threshold is applied, on the right. You can see that the edges of the stars becoming ragged, losing some signal to the background noise, and the dimmest star (upper left) is broken up into small pieces.

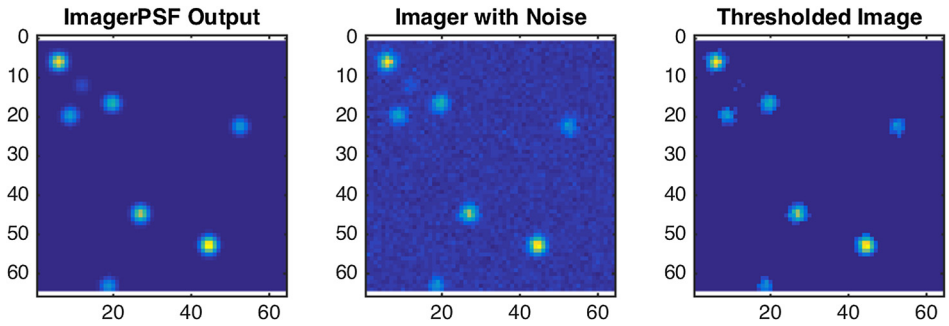


Figure K.3 Effect of thresholding a simulated image. Thresholding, on the right, removes the noise.

K.3.2 Creating a star blob

The blobification routine performs a search on the pixels left after thresholding. If pixels are adjacent to any other remaining pixels, they are added to the blob. A blob consists of the following parameters:

- Bounding box: top/bottom row indices, left/right column indices;
- List of pixels in the blob;
- Location and value of brightest pixel in the blob.

For each pixel in the thresholded image, if it is adjacent to one of the existing blobs, it is added to that blob's list. If it is brighter than the brightest pixel, the brightest pixel is updated. If the pixel is not in any existing blobs, a new blob is created.

Example blobs are shown in Fig. K.4. The white box is the bounding box.

K.3.3 Center-of-mass

The center-of-mass algorithm is a simple weighted average of the pixels in a region of interest centered on a bright pixel. In the most general form of the algorithm, a local noise floor can be computed from the ring of pixels surrounding the region of interest. This step can be skipped if the image is already thresholded as in Section K.3.1.

The center-of-mass algorithm has the following steps:

1. Input is a square region of interest (ROI) centered around the brightest pixel.
2. Calculate a noise threshold from the pixels surrounding the ROI (optional).
3. Compute the total intensity of the ROI (above the noise floor).
4. Compute the centroid by weighting each pixel by its intensity.

In a tracking implementation, the first step is not needed; the center pixels would be an input using the previous frame along with estimation of the satellite's rotation. The first step is therefore part of the lost-in-space (LIS) solution. The region of interest and the edge used for calculating the noise floor are shown in Fig. K.5.

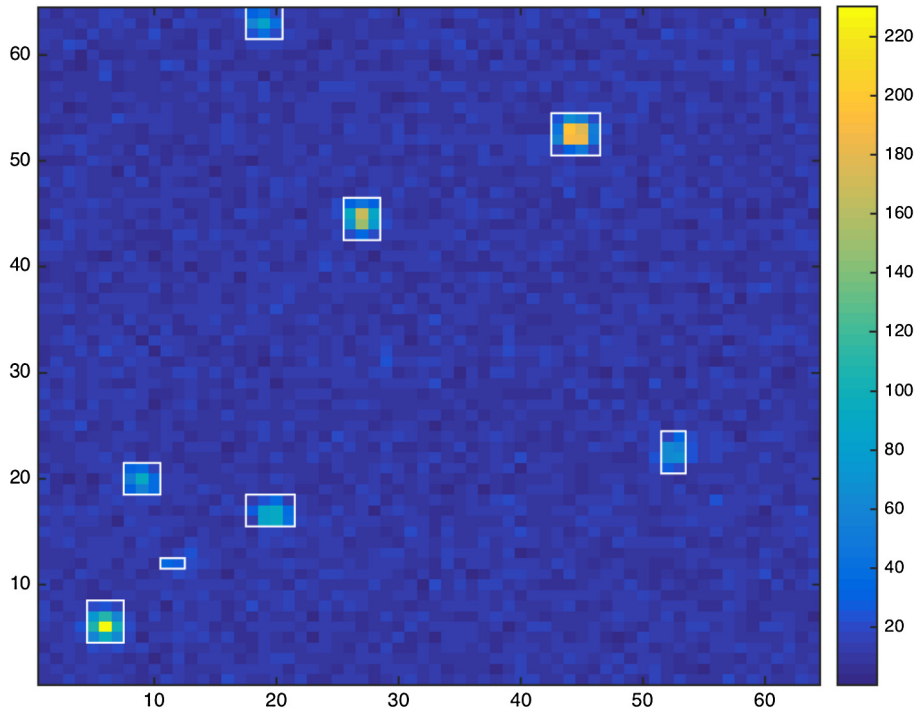


Figure K.4 Blobify routine results. Seven blobs, which are potential stars, are identified.

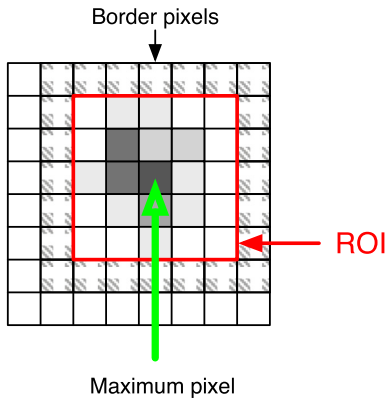


Figure K.5 Region of interest with border.

The noise floor η is an average of the N pixels around the edge of the region of interest (ROI). j is the column value of the pixel and i is the row value. This may be substituted with a constant noise value or skipped if the noise has been subtracted in a

prior step,

$$\eta = \sum_{edge} p(j,i)/N$$

The total intensity is the sum of the pixels in the ROI after subtracting the noise threshold.

$$I = \sum_i \sum_j p(j,i) - \eta$$

The x and y values of the centroid are then calculated using a center-of-mass calculation of the pixels.

$$x = \frac{\sum\sum j(p(j,i)-\eta)}{I} + 0.5$$
$$y = \frac{\sum\sum i(p(j,i)-\eta)}{I} + 0.5$$

where x and y are defined with the origin at the corner of the image. The origin needs to be transformed to the center of the frame before the centroids can be used by the star-identification algorithm.

K.4. Star identification

A pyramid star ID algorithm is used for star identification. Identifying a pyramid of stars virtually assures a correct lost-in-space identification, and allows the algorithm to exit the search once a pyramid is found. The algorithm first uses triad checking to find a candidate triad, then checks the remaining measurements for a self-consistent pyramid match, i.e., a unique match with all three stars in the triad. This dramatically reduces the processing time on a new image, from 0.2 second to identify stars from 15 centroids to only 0.02 seconds (in MATLAB®). This correlates well with the reference for a fast pyramid technique [2]. It has been tested on orbit on the High Energy Transient Explorer (HETE) spacecraft.

K.4.1 Catalog processing

The star-catalog processing is shown in Fig. K.6. We start with the Hipparcos catalog. Stars are selected that are above the minimum desired magnitude, which is based on the camera field-of-view and planned exposure time. Then, any stars that will be so close together as to be within the same region of interest used for centroiding, must be removed. The field-of-view is then used to generate the near matrix of star pairs, or all pairs of stars that are close enough to appear together in an image. Finally, the resulting star pairs are sorted and the k-vector parameters stored for fast lookup.

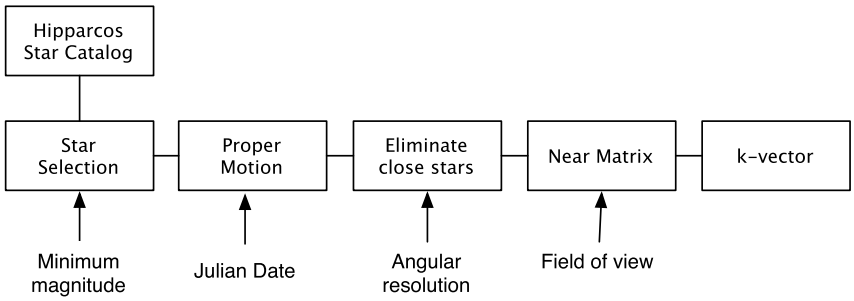


Figure K.6 Star-catalog processing. This reduces the catalog to a manageable size and makes star identification easier.

The catalog can be further optimized by restricting the number of stars kept in the densest regions of the sky [3]. If there are more than a set number of stars in a field-of-view (FOV) region, for example 12 stars, then only the brightest 12 are kept. This reduces the number of star pairs in the near matrix and makes star identification faster. The Hipparcos catalog has 13 943 stars with a minimum visual magnitude of 7, which would have a minimum of 19 stars in a 15 degree field-of-view. A magnitude of 6.5 results in 7982 stars and a minimum of 9 in the field-of-view. If we optimize a catalog using magnitude 7 stars, we can have a catalog with just 4436 stars and a minimum of 11 in the field-of-view (using an increment of 10% of the field-of-view to scan the sky). This is shown in Fig. K.7; note that the shape of the milky way is not readily apparent and that the star density is fairly uniform.

K.4.2 Sorting star pairs with *k*-vector

The *k*-vector range search technique allows a range of values from the near matrix to be extracted without searching, by using indexing. The star angles of the star pairs in the near matrix are sorted in ascending order, producing an array of star angles s, and an additional integer index value is assigned to each pair. A slope and intercept is calculated for an equivalent straight line through the data. The indices of s can then be calculated for a given star-angle range and all values within the range easily retrieved. Practically, the index k gives the number of elements in the angle array below the value dictated by the equivalent line.

The equation of the straight line connecting the minimum and maximum angles, extended by a small delta ξ is

$$z(x) = mx + b$$

where

$$m = \frac{s_{max} - s_{min} + 2\xi}{n - 1} \tag{K.9}$$

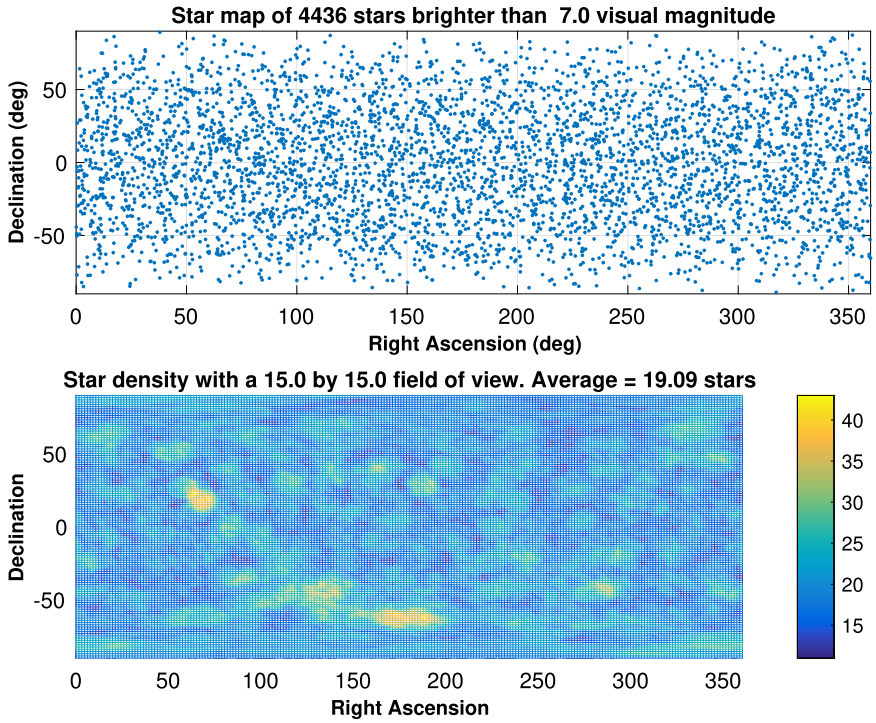


Figure K.7 Reduced star catalog. The field-of-view is used to reduce the catalog size. Only the brightest stars in a given region are kept.

$$q = \gamma_{min} - m - \xi \tag{K.10}$$

$$\xi = \epsilon \max[s_{min}, s_{max}] \tag{K.11}$$

and ϵ is the relative machine precision, and s_{min} and s_{max} are the minimum and maximum values of the input data s. That is, z is the line connecting the points $[1, s_{min} - \zeta]$ and $[n, s_{max} + \zeta]$ where there are n star-pair angles in s.

The integer vector k is constructed as

$$k(i) = j \quad \text{if the } j \text{ index satisfies} \quad s(j) \leq z(i) < s(j+1)$$

starting with $k(1) = 0$.

The evaluation of two indices k_{start} and k_{end} identifying the data within a given range $[\gamma_a, \gamma_b]$ involves calculating the floor and ceiling values, j_a and j_b,

$$j_a = \left\lfloor \frac{\gamma_a - q}{m} \right\rfloor \quad \text{and} \quad j_b = \left\lceil \frac{\gamma_b - q}{m} \right\rceil$$

With these indices it is possible to look up the indices of s,

$$k_{start} = k(j_a) + 1 \quad \text{and} \quad k_{end} = k(j_b)$$

which guarantees that

$$s(k_{start}) < \gamma_a \quad \text{and} \quad \gamma_b < s(k_{end})$$

Therefore the possible matching star pairs are those in the matrix of pairs between k_{start} and k_{end}. Note that it is possible for the range to include extra points beyond the inputs, which are still within the range of the closest line indices; these can be eliminated with a linear search.

The script KVectorDemo produces a demo with the plots in Fig. K.8. This is for a simple set of random numbers on the interval (0,1). The plot on the left shows the steps in k, in a stair plot, as compared against a linear progression. For example, at index 4, k jumps from two to four, indicating that there are four data points with values below the line value at that index. On the right, the data is printed against the straight-line fit to the endpoints, with a range calculated of 0.5 to 0.75. The range is marked with the magenta solid lines. The data points that are circled are the values returned as in the desired range. We can see in this particular example that one data point below 0.5 was returned, because it is also above the value of the closest line point, 0.4642. The line points bracketing the range are indicated by the black dashed lines.

When using k-vector with measured star pairs, the angle range will be the measured angle $\theta \pm \delta$. This should be a high confidence interval such as 4σ. This could be a constant value for the entire pixel array or calculated based on the star locations or angle.

K.5. Fine centroiding

The concept for fine centroiding involved a numerical fit of an integrated 2D point-spread function (PSF) against the pixel values for each star. However, the defocused Airy PSF for a single wavelength does not resemble the PSF for the full-wavelength spectrum of a star, which in fact resembles a Gaussian. A Gaussian can be readily fit to the pixel image in separate X and Y directions, in a procedure known as digital centering of the marginal distributions [4]. This algorithm is used in astrometry. This performs very well against the numerical fit of a 2D Gaussian to a simulated image smeared with Poisson noise. This effectively corrects errors from the coarse centroiding that are larger as stars are located more to the outside corners of the pixel.

The digital centering means fitting a Gaussian to the pixel sums in each coordinate axis of the focal plane. This can be accomplished using a Newton–Gauss method (least squares) with analytic derivatives in just 2 or 3 iterations. The fit gives a measure of both the star-centroid location and the PSF width (Gaussian standard deviation σ). This

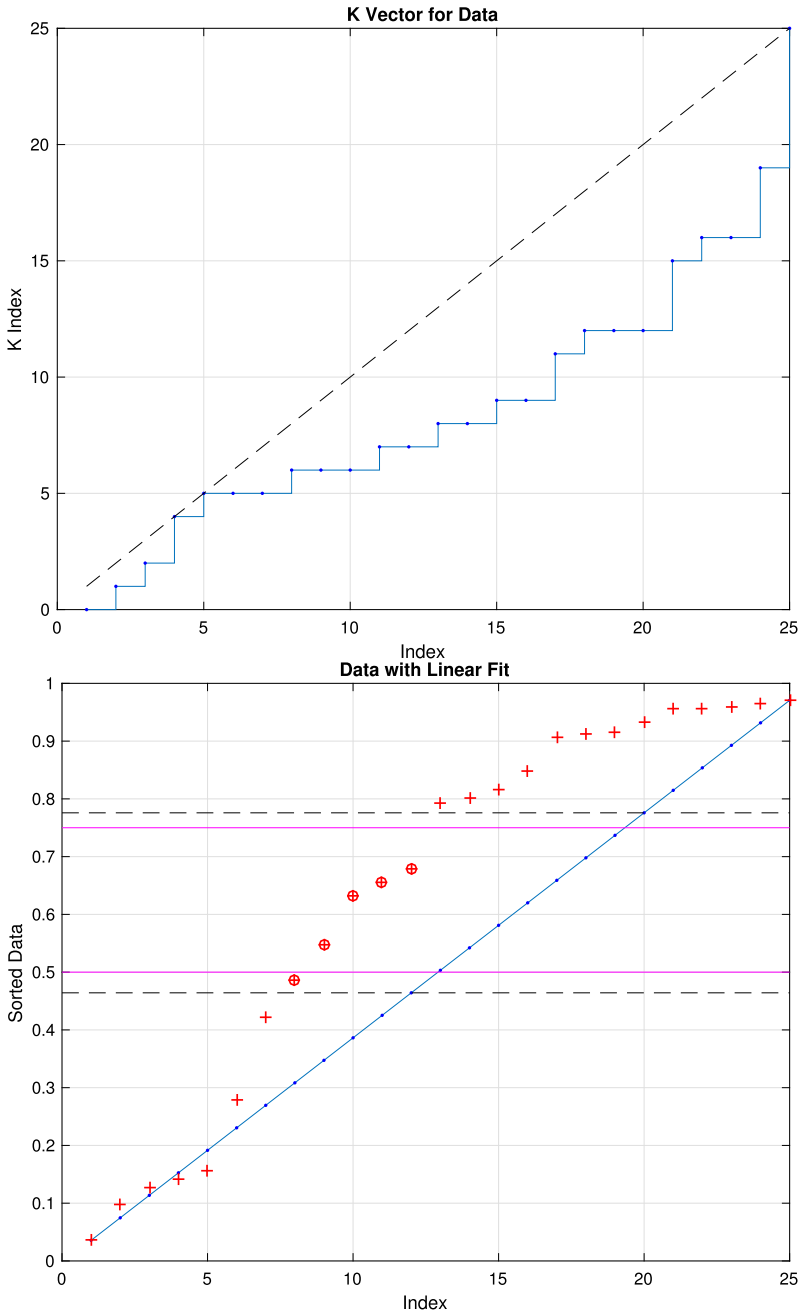


Figure K.8 K-vector example in MATLAB. The plot on the left shows the steps in k. The right plot shows the data display against a straight-line fit.

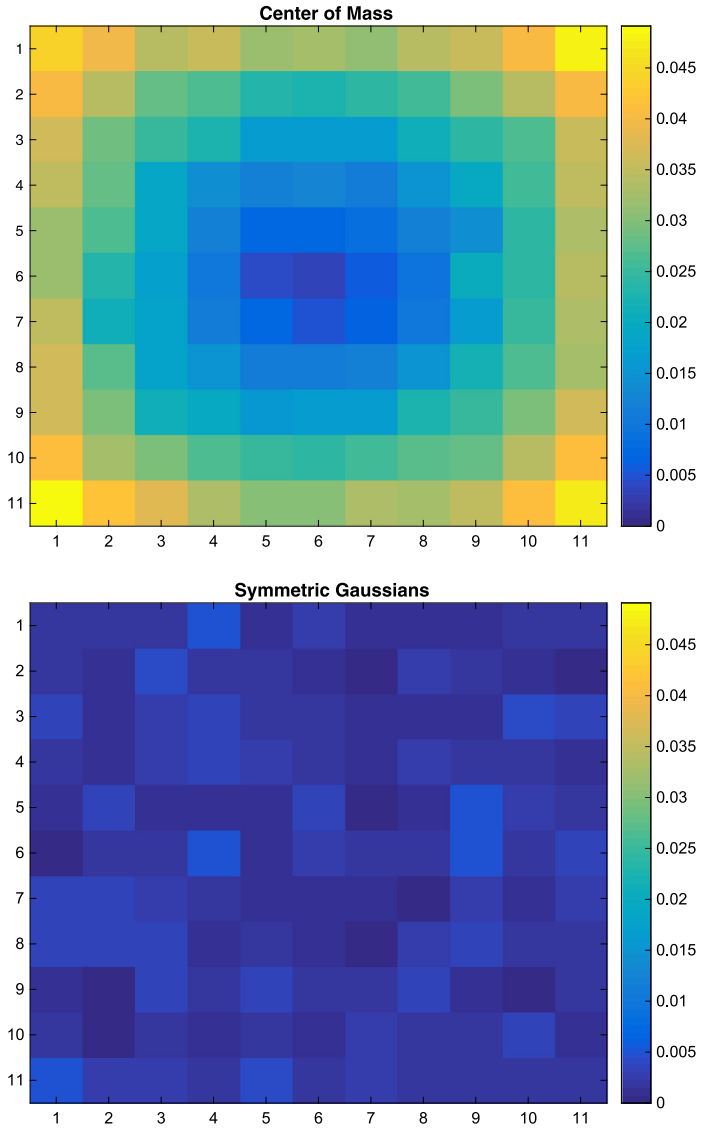


Figure K.9 Center-of-mass centroiding compared against Gaussian fitting to the marginal distribution when the centroid moves throughout the pixel. The scales of the color bars in both images are the same.

is essentially the same method used to fit the sky background to the image histogram in Section K.3.1. In implementation, it is as fast as the center-of-mass method, while eliminating errors when the centroid is towards the corner of the frame. The following two plots in Fig. K.9 show a comparison of the fractional centroid error as the centroid

is moved throughout a pixel, with the Gaussian centering showing a clear improvement. This is for a PSF simulated using a Gaussian and smeared with Poisson noise.

Fig. K.10 shows an example of the digital centering method. The sums of the values in each direction are shown on the bar chart. The Gaussian fit is shown on top in red. Note that the height of the fit is greater than the height of the data. The source pixelmap is shown on the right.

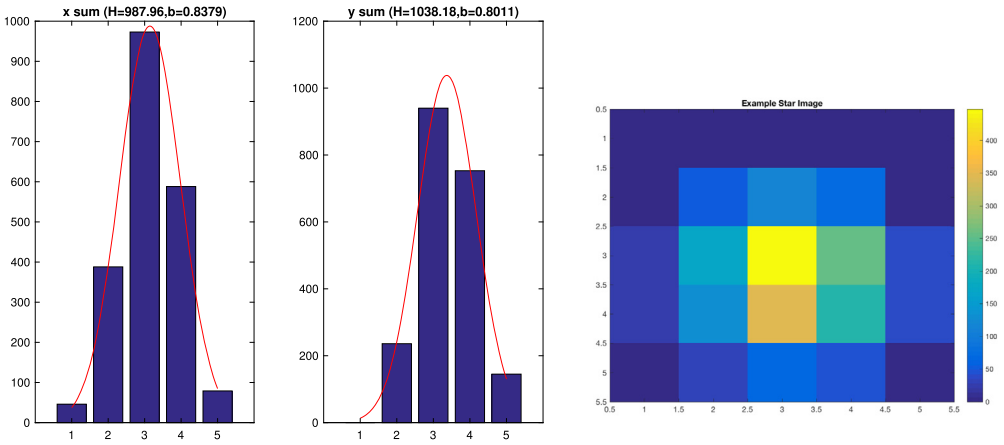


Figure K.10 Digital centering example.

References

[1] A. Bijaoui, Sky background estimation and application, Astronomy & Astrophysics 84 (1–2) (April 1980) 81–84.
[2] D. Mortari, M. Samaan, C. Bruccoleri, J. Junkins, The pyramid star identification technique, Navigation 51 (3) (September 2004) 171–183.
[3] J.D. Vedder, Star trackers, star catalogs, and attitude determination: probabilistic aspects of system design, Journal of Guidance, Control, and Dynamics 16 (3) (May–June 1993).
[4] R.C. Stone, A comparison of digital centering algorithms, The Astronomical Journal 97 (4) (April 1989).

APPENDIX L

Magnetic-hysteresis damping

This chapter discusses magnetic-hysteresis damping. This is an approach often used in lower-cost satellites. When used with a magnetic torque rod they can dissipate energy without any moving parts and without any additional power, aside from the power for the torque rod.

L.1. Magnetic-hysteresis damper model

Hysteresis dampers are passive and act continuously to damp angular rates. The flux vector through the rod, $\boldsymbol{B}$, can be used to calculate the rod's magnetic dipole, $\boldsymbol{M_D}$, from

$$\boldsymbol{M_D} = \boldsymbol{B}V \tag{L.1}$$

where V is volume of the hysteresis rods. The flux is a function of the applied magnetic field $\boldsymbol{H}$, the vacuum permeability $\mu_0 = 4\pi \times 10^{-7}$ H/m, and the apparent permeability of the rod μ_r'. μ_r' is crucial in determining the effect of finite geometric constraints on the magnetic properties; it is a function of the true material permeability μ_r and the demagnetizing factor, N_d,

$$\boldsymbol{B} = \mu_0 \mu_r' \boldsymbol{H} \tag{L.2}$$

$$\mu_r' = \frac{\mu_r(\boldsymbol{H})}{1 + N_d \mu_r(\boldsymbol{H})} \tag{L.3}$$

The demagnetizing factor N_d is given by [1], as

$$\begin{aligned}
\pi N_d = {} & \frac{b^2 - c^2}{2bc} \ln\left(\frac{\sqrt{a^2+b^2+c^2} - a}{\sqrt{a^2+b^2+c^2} + a}\right) + \frac{a^2 - c^2}{2ac} \ln\left(\frac{\sqrt{a^2+b^2+c^2} - b}{\sqrt{a^2+b^2+c^2} + b}\right) \\
& + \frac{b}{2c} \ln\left(\frac{\sqrt{a^2+b^2} + a}{\sqrt{a^2+b^2} - a}\right) + \frac{a}{2c} \ln\left(\frac{\sqrt{a^2+b^2} + b}{\sqrt{a^2+b^2} - b}\right) + \frac{c}{2a} \ln\left(\frac{\sqrt{c^2+b^2} - b}{\sqrt{c^2+b^2} + b}\right) \\
& + \frac{c}{2b} \ln\left(\frac{\sqrt{a^2+c^2} - a}{\sqrt{a^2+c^2} + a}\right) + 2\arctan\left(\frac{ab}{c\sqrt{a^2+b^2+c^2}}\right) + \frac{a^3 + b^3 - 2c^3}{3abc} \\
& + \frac{a^2 + b^2 - 2c^2}{3abc} \sqrt{a^2+b^2+c^2} + \frac{c}{ab}\left(\sqrt{a^2+c^2} + \sqrt{b^2+c^2}\right) \\
& - \frac{\left(a^2+b^2\right)^{3/2} + \left(b^2+c^2\right)^{3/2} + \left(c^2+a^2\right)^{3/2}}{3abc}
\end{aligned}$$

where the dimensions of the magnet are $\pm a$, $\pm b$, and $\pm c$, with the axis along c being most-closely aligned with the magnetic field. Based on this result, [2] gives an apparent rod saturation flux

$$B_s' = \frac{\mu_r' B_s}{\mu_r^m} \tag{L.4}$$

The theoretical calculation bounds of μ_r' and B_s' are upper limits themselves, e.g., they assume perfect manufacturing. Any manufacturing imperfections will result in less efficiency, which translates into a lower range of apparent permeability and therefore a lower range of saturation flux. This effect is discussed further in [2].

The energy dissipated per hysteresis cycle, D, by a damper is

$$D = V \oint \boldsymbol{H} d\boldsymbol{B} \tag{L.5}$$

To find $\oint HdB$, which is the area within the permeable rod hysteresis loop, as well as establish the dynamics for the magnetic hysteresis torque, we use the hysteresis model based on [3], [4].

The major hysteresis loop is

$$\boldsymbol{B} = \frac{2B_S}{\pi} \tan^{-1}[k(\boldsymbol{H} \pm H_C)] \tag{L.6}$$

where B_S is the saturation flux density, H_C is the coercive force, and k is a shaping factor

$$k = \frac{1}{H_C} \tan\left(\frac{\pi B_R}{2B_s}\right) \tag{L.7}$$

where B_R is the remanence. The slope is

$$\left(\frac{d\boldsymbol{B}}{d\boldsymbol{H}}\right)_{ml} = \frac{2kB_S}{\pi} \cos^2\left(\frac{\pi \boldsymbol{B}}{2B_S}\right) \tag{L.8}$$

The left and right boundary values for H are

$$H_L = \hat{H} - H_C \tag{L.9}$$

$$G = \frac{H - H_L}{2H_C} \tag{L.10}$$

$$H_R = \hat{H} + H_C \tag{L.11}$$

$$G = \frac{H + H_R}{2H_C} \tag{L.12}$$

where the term $\hat{H}$ is defined as follows:

$$\hat{H} = \frac{1}{k} \tan\left(\frac{\pi \boldsymbol{B}}{2B_S}\right) \tag{L.13}$$

The slope is

$$\frac{d\boldsymbol{B}}{d\boldsymbol{H}} = \mathrm{G}^2 \left(\frac{d\boldsymbol{B}}{d\boldsymbol{H}}\right)_{ml} \tag{L.14}$$

and

$$\frac{d\boldsymbol{B}}{dt} = \frac{d\boldsymbol{B}}{d\boldsymbol{H}} \frac{d\boldsymbol{H}}{dt} \tag{L.15}$$

The material hysteresis loop is shown in Fig. L.1.

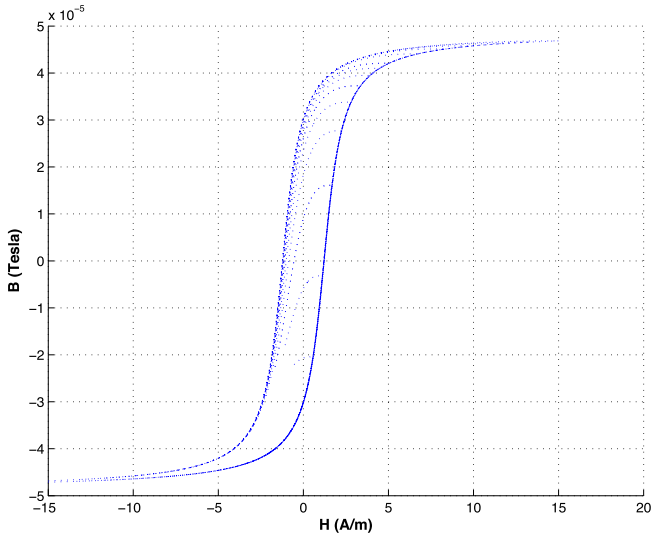


Figure L.1 Hysteresis for Permalloy C. The dotted lines are minor loops.

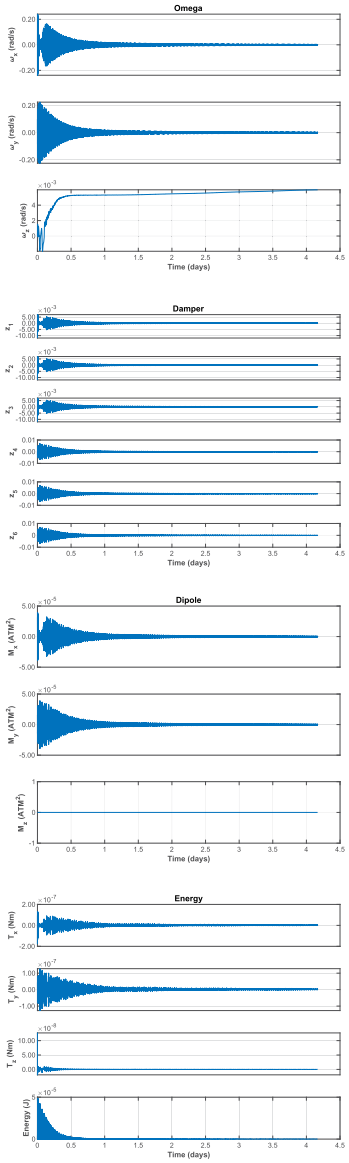
Integrating the major hysteresis loop over the maximum range of Earth's magnetic field at our altitude and inclination, which is ±36.1 A/m, and inserting H_L and H_R into (L.6) to get the left-hand and right-hand curves, respectively, yields 2.44×10^{-5} N/m^2 $\leq \oint HdB \leq 2.30 \times 10^{-4}$ N/m^2.

L.2. Energy-dissipation analysis

The average power dissipated is [5]

$$\dot{E} = -Df \tag{L.16}$$

```
%% Constants
secInDay      = 86400;
mu0           = 4e-7*pi;

%% User inputs
dateStart     = [2023 6 1 0 0 0];
tDuration     = 100*3600;
q0            = [1;0;0;0];
torqueD       = [0;0;0]*1e-7;
tPulse        = [1 2]*100000;
dT            = 2;
omega0        = randn(3,1)*0.0001;
mRDamping     = true;
d             = RHSRigidBodyWithDamping;
d.dipole      = [0;0;1];
[d.inertia, d.mass]      = InertiaCubeSat('1U');
cDamp         = 0.0001;

%% Start date
jDStart = Date2JD(dateStart);

% Orbit
[el,jD0] = ISSOrbit;
el(6)     = el(6) + (jDStart-jD0)*secInDay*2*pi/
        Period(el(1));
[r,v]     = El2RV(el);

% Time vector
n  = ceil(tDuration/dT);
t  = linspace(0,tDuration,n);
jD = jDStart + t/secInDay;

if( mRDamping )
  d.dampingType = 0;
  d.dampingData.Br = 0.004;
  d.dampingData.Bs = 0.025;
  d.dampingData.Hc = 12;
  % Damper rod unit vectors
  d.dampingData.u   = [0 0 0 1 1 1;1 1 1 0 0 0;0
      0 0 0 0 0];
  % Dimensions are radius 1 mm by 95 mm
  d.dampingData.v = pi*0.001^2*0.095*ones(1,size(
      d.dampingData.u,2));
  d.dampingFun = @RHSHysteresisDamper;
  uECI = QTForm(q0,d.dampingData.u);
  [bI,bIDot] = BDipole(r,jDStart,v);
  hMag = Dot(uECI,bI    )/mu0;
  hMagDot = Dot(uECI,bIDot)/mu0;
  z = BFromHHysteresis( hMag, hMagDot, d.
      dampingData )';
else
  % constant damping
  d.dampingType      = 1;
  z = zeros(6,1);
  d.dampingData = cDamp;
end
x = [r;v;q0;omega0;z];
xP = zeros(20,n);

%% Simulation loop
for k = 1:n

  % Plotting
  [~,p] = RHSRigidBodyWithDamping( x, t(k), d );
  omega = x(11:13);
  energy = 0.5*omega'*d.inertia*omega;
  xP(:,k) = [x(7:end);p.torqueDipole;p.
      torqueDamper;energy];

  % Disturbance pulse for testing
  if( t(k) > tPulse(1) && t(k) < tPulse(2))
    d.torque = torqueD;
  else
    d.torque  = [0;0;0];
  end

  % Integrate
  x = RK4(@RHSRigidBodyWithDamping,x,dT,t(k),d);
end

%% Plotting
yL  = [d.states(:)' {'M_x␣(ATM^2)'} {'M_y␣(ATM^2)
    '} {'M_z␣(ATM^2)'} ...
{'T_x␣(Nm)'} {'T_y␣(Nm)'} {'T_z␣(Nm)'} {'Energy␣(
    J)'}];

TimeHistory(t,xP( 5: 7,:),yL(11:13),'Omega');
TimeHistory(t,xP( 8:13,:),yL(14:19),'Damper');
TimeHistory(t,xP(14:16,:),yL(20:22),'Dipole');
TimeHistory(t,xP(17:20,:),yL(23:26),'Energy');
Figui
```



Example L.1: Magnetic-hysteresis damping. The energy drops towards zero.

The damper torque is [3]

$$T = \frac{vbu^{\times} b_E}{\mu_0} \tag{L.17}$$

where b is the damper axial flux density, v is the volume of the damper, b_E is the Earth's magnetic field, u is the damper unit vector in the body frame, and μ_0 is the permeability of free space.

Example L.1 shows magnetic-hysteresis damping. The spacecraft has a 1-ATM^2 dipole along the z-axis. The script starts with a random angular rate. It takes several hours to damp the angular rates. The rates transverse to the magnetic torque rod go to zero.

References

[1] A. Aharoni, Demagnetizing factors for rectangular ferromagnetic prisms, Journal of Applied Physics 83 (6) (March 1998).
[2] F. Santoni, M. Zelli, Passive magnetic attitude stabilization of the UNISAT-4 microsatellite, Acta Astronautica 65 (5–6) (2009) 792–803.
[3] R.R. Kumar, D.D. Mazanekand, M.L. Heck, Simulation and shuttle hitchhiker validation of passive satellite aerostabilization, Journal of Spacecraft and Rockets 32 (5) (September–October 1995).
[4] T.W. Flatley, D.A. Henretty, A magnetic hysteresis model, in: K.R. Hartman (Ed.), Flight Mechanics/Estimation Theory Symposium 1995, May 1995, pp. 405–415.
[5] V. Francois-Lavet, Study of passive and active attitude control systems for the OUFTI nanosatellite, Master's thesis, University of Liège, 2009–2010.

APPENDIX M

Machine intelligence

M.1. Space story

NASA's Deep Space 1 had an onboard "intelligent" control system that could do the planning. It was the first interplanetary mission to use an ion engine. It visited the asteroid 9969 Braille and the comet Borrelly. It was one of the first spacecraft to have Artificial-Intelligence software.

M.2. Introduction

Machine intelligence is a field in computer science where data is used to predict, or respond to, future data. It is closely related to the fields of pattern recognition, computational statistics, and Artificial Intelligence. The data may be historical or updated in real time. Machine learning is important in areas like facial recognition, spam filtering, and other areas where it is not feasible, or even possible, to write algorithms to perform a task. In many cases tedious work that had been done by humans can now be done by machines. Machine-learning algorithms can also generate new data, such as music or stories, by use of massive learning of sources.

Spacecraft are a natural fit for machine intelligence because all satellites need a high degree of autonomy. Even satellites that are under constant ground monitoring, such as geosynchronous communications satellites, benefit from high degrees of autonomy because problems can be very expensive to fix. In addition, operator time is expensive. For deep-space missions, communication time delays make autonomy even more important. Deep-space operations are also expensive, costing tens of millions of US dollars a year. Autonomous systems can reduce these costs. Most spacecraft today incorporate autonomy using boolean logic, the familiar if-else-then type of statements. The parameters for the logic are usually determined before launch. For example, boolean logic might check attitude errors measured by the sensors, and if a value is exceeded switch the spacecraft into a Sun-safe mode. Another example would be logic that checks the electrical current draw for a sensor or actuator, and switches devices if no current is detected.

This appendix will first give an overview of machine learning. The focus of the remaining parts of the appendix will be on deep learning. The first section then derives the Exclusive-OR (XOR) example showing how a neural network can replicate the XOR. Exclusive-OR is a fundamental operation used in computers with two inputs and one output. The output can take one of two states, depending on the four inputs.

An example of orbit determination using a neural network is given. Another example of using a neural network to get roll-and-pitch measurements from a static Earth sensor is described. An expert system will be demonstrated. Finally, an example of using reinforcement learning for an attitude maneuver with reaction wheels will be presented. The latter will be compared with an optimal maneuver using downhill Simplex.

M.3. Branches of machine intelligence

Machine learning as described above is the collecting of data, finding patterns, and doing useful things based on those patterns. We expand machine learning to include adaptive and learning control. These fields started independently but now are adapting technology and methods from machine learning. All control systems deal with uncertainty. For example, gain and phase margins are a measure of the tolerance a system has to uncertainty. Adaptive control systems go a step further and learn from their experiences. This is done via an algorithm. With machine learning, we usually forego creating an algorithm in advance and let the system find the method of relating inputs to outputs.

Fig. M.1 shows how we organize the technology of machine learning into a consistent taxonomy. You will note that the title encompasses three branches of learning; the whole subject area is called "Autonomous Learning". Which means, learning without human intervention during the learning process. This book is not solely about "traditional" machine learning. Other, more specialized books focus on any one of the machine-learning topics. For example, optimization is a tool used for trajectory planning and other spacecraft applications. The optimization software is often part of deep-learning training. It is used to find the optimal weights for the neurons for the neural networks.

There are three categories under Autonomous Learning. The first is *Control*. Feedback control is used to compensate for uncertainty in a system or to make a system behave differently from how it would normally behave. If there was no uncertainty you would not need feedback. For example, if you are kicking a ball at a teammate who is running, assume for a moment that you know exactly where the teammate will be at any time. This is the point of having predefined plays in sports. You know exactly where your teammate should be at a given time, so you can close your eyes, count, and just kick the ball to that spot. Assuming your teammate has good feet, you would have a 100% reception rate! More realistically, you watch the player, estimate their speed, and kick the ball. You are applying feedback to the problem. As stated, this is not a learning system. However, if now you practice the same play repeatedly, look at your success rate and modify the mechanics and timing of your kick using that information, you would have an adaptive control system, the box second from the top of the control list. Learning in control takes place in adaptive control systems and also in the general area of system identification.

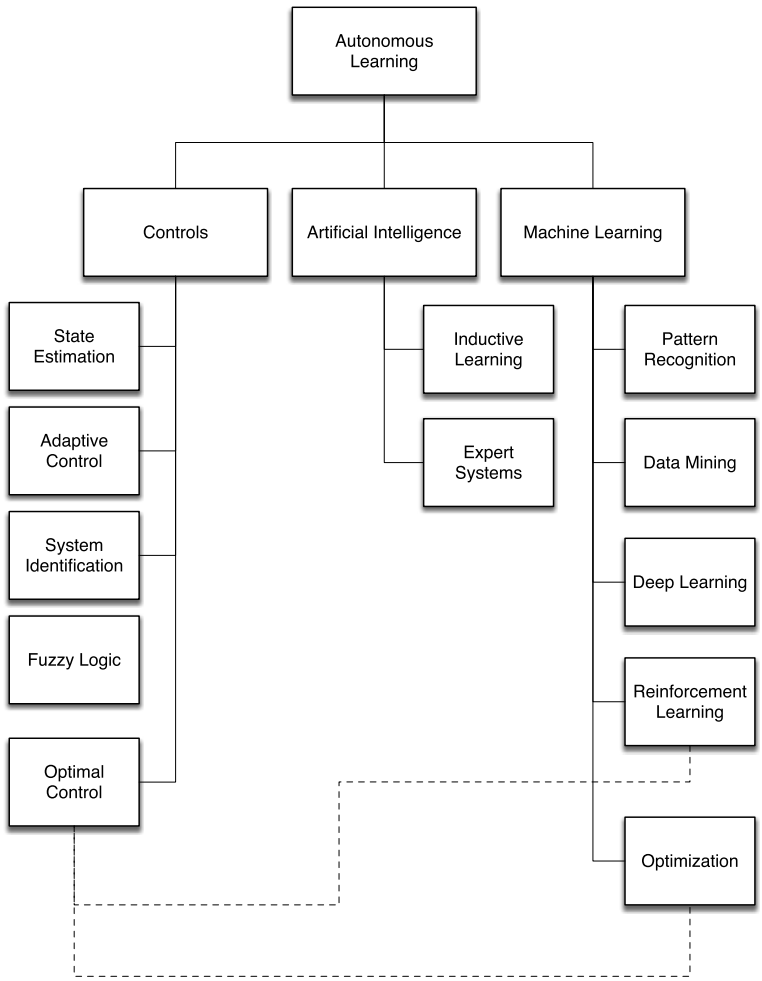


Figure M.1 Taxonomy of machine learning.

System identification is learning the parameters of the system. By system, we mean the data that represents anything and the relationships between elements of that data. For example, a particle moving in a straight line is a system defined by its mass, the force on that mass, its velocity, and its position. The position is the integral of the velocity over time and the velocity is the integral of the acceleration, which is the force divided by the mass. The concept of state has been discussed previously. Characteristics of a system can be compartmentalized into states and parameters. In general, the latter do not change in response to external influences.

Optimal control may not involve any learning. For example, full-state feedback produces an optimal control signal but does not involve learning. In full-state feedback, the combination of model and data tells us everything we need to know about the system. However, in more complex systems it is not possible to measure all the states or to know the parameters perfectly so some form of learning is needed to produce "optimal" or the best possible results.

The second category is what is usually considered true *Machine Learning*. This is the use of data to produce behavior that solves problems. Much of its background comes from statistics and optimization. The learning process may be done once in a batch process or continually in a recursive process. For example, in a stock-buying package, a person might have processed stock data for several years and used that to decide which stocks to buy. That software might not have worked well during a financial crash that had never been observed and for which there is no data. A recursive program would continuously incorporate new data. Pattern recognition and data mining fall into this category. Pattern recognition is used to find patterns in images. For example, the early AI Blocks World software could identify a block in its field of view. It could find one block of a particular color in a pile of blocks and determine its location. Data mining is taking large amounts of data and looking for patterns. For example, taking stock-market data and identifying companies that have strong growth potential. Classification techniques and fuzzy logic are also in this category.

The third category of autonomous learning is *Artificial Intelligence*. Machine learning traces some of its origins to Artificial Intelligence. Artificial Intelligence is the area of study whose goal is to make machines reason. While many would say the goal is "think like people" this is not necessarily the case. There may be ways of reasoning that are not similar to human reasoning but are just as valid. In the classic Turing test, Turing proposes that the computer only needs to imitate a human in its output to be a "thinking machine", regardless of how those outputs are generated. A system passes the Turing test if a human cannot tell if the responses are from a machine or another human being. In any case, intelligence generally involves learning so learning is inherent in many Artificial-Intelligence technologies such as inductive learning and expert systems. Our diagram includes the two techniques of inductive learning and expert systems.

M.4. Stored command lists

Spacecraft are often operated by commands sent from the ground station. A sequence of commands is usually required to perform a spacecraft operation. For example, to start a station-keeping maneuver with hydrazine thrusters the commands might be

1. Turn on the catalyst-bed heaters for the hydrazine thrusters.
2. Turn on the thrusters.
3. Turn on the rate gyros.

4. Switch to station-keeping attitude control mode.
5. Turn on the delta-V thrusters.

Each command is sent in sequence. Positive verification that the command was loaded in the spacecraft is usually required before sending the next command. This is known as handshaking and requires that telemetry be sent from the spacecraft and received at the ground station,

A degree of spacecraft autonomy can be achieved by storing these lists onboard the spacecraft. In this way a complex sequence can be performed with a single ground command. This reduces the workload on the ground station. In the case of the station-keeping sequence, the ground would initiate the sequence and then initiate the start of the burn. In this way the operators can make sure that all of the commands were executed properly. The next step would be to have the spacecraft check to see if a command was properly implemented. For example, it could check the current draw on the gyros before proceeding to the next step. A command-execution engine would eliminate the need for writing custom logic for each command list. The next step is to implement an expert system for performing command sequences. The expert system would be able to assemble a command list from a goal for the operation.

M.5. Deep Space 1

The Remote Agent experiment on the NASA Deep Space 1 mission provided a degree of autonomy on the spacecraft [1]. The spacecraft is shown in Fig. M.2. The mission also tested other technologies such as autonomous navigation and ion propulsion.

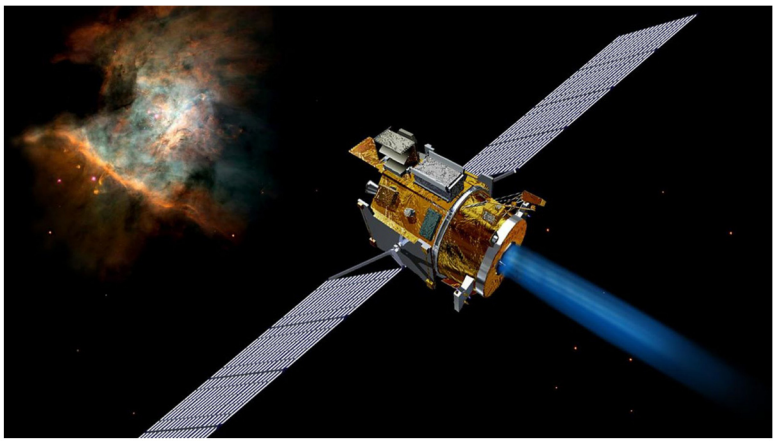


Figure M.2 The Deep Space 1 spacecraft. Image courtesy of NASA.

The architecture is shown in Fig. M.3. Real-time software is the flight software system that does conventional operations. The Remote Agent is managing that system,

much like ground controllers would do. The experiment manager operates the Remote Agent to do a series of experiments as part of the test program. The planner/scheduler maintains a database of goals for the mission.

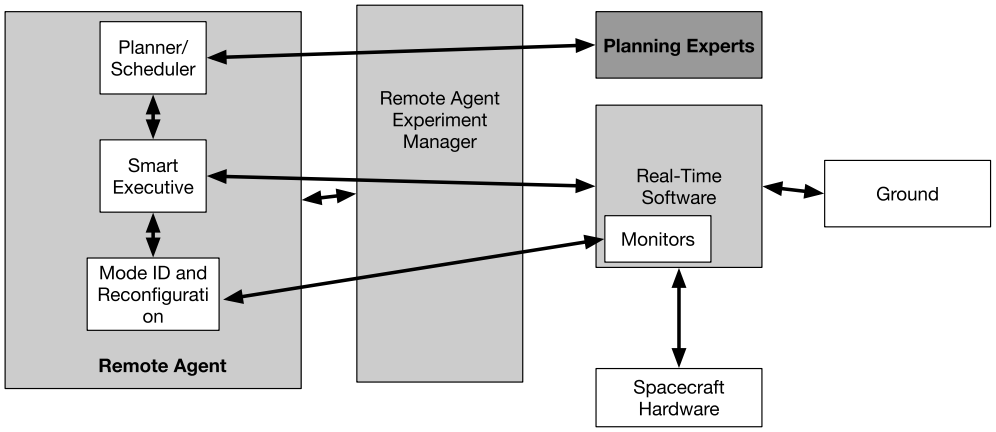


Figure M.3 Deep Space 1 autonomous flight system [2].

The Remote Agent enabled goal-based planning and robust failure recovery. The spacecraft was launched in October 1998 and the Remote Agent experiment was run in May of 1999. The levels of autonomy were

1. Single low-level command execution;
2. Time-stamped commands;
3. Single goal achievement with autorecovery;
4. Model-based state estimation and error detection;
5. Scripted plans with dynamic task decomposition;
6. Onboard back-to-back plan generation, execution, and recovery.

The first two are standard commanding. The others allowed the operators to set goals and have the system achieve those goals subject to operational constraints. For example, an operator could ask the spacecraft to take a picture and the system would make sure it could accomplish that goal. Default goals were also a part of the system, i.e., point an antenna at the Earth if the system had no other ongoing tasks. The planning was robust as it could automatically replan based on new information.

The Remote Agent Experiment was written in the Lisp programming language. Lisp is one of the first Artificial-Intelligence languages [3]. This presented challenges both because it was running on a processing-limited radiation-hard processor and had to interact with the C flight software. The experiment rigorously tested the software [4]. The goal was to produce a package that could be used on future missions.

M.6. Neural networks

Neural networks, or neural nets, are a popular way of implementing machine "intelligence". The idea is that they behave like the neurons in a brain. In this section, we will explore how neural nets work, starting with the most fundamental idea of a single neuron and working our way up to a multilayer neural net.

The simplest problem is a single neuron with two inputs. This is shown in Fig. M.4. This neuron has inputs x_1 and x_2, a bias b, weights w_1 and w_2, and a single output z. The activation function σ takes the weighted input and produces the output. In this diagram, we explicitly add icons for the multiplication and addition steps within the neuron, but in typical neural-net diagrams such as Fig. M.4, they are omitted.

$$z = \sigma(y) = \sigma(w_1 x_1 + w_2 x_2 + b) \tag{M.1}$$

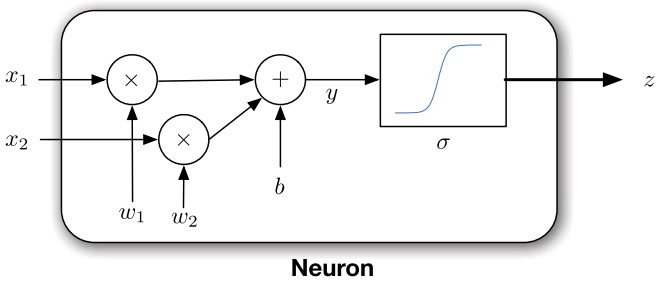


Figure M.4 A two-input neuron.

Let us compare this with a real neuron as shown in Fig. M.5. A real neuron has multiple inputs via the dendrites. Some of these branches can connect to the cell body through the same dendrite. The output is via the axon. Each neuron has one output. The axon connects to a dendrite through the synapse. Signals pass from the axon to the dendrite via a synapse.

There are numerous commonly used activation functions. Four are:

$$\sigma(y) = \tanh(y) \tag{M.2}$$

$$\sigma(y) = \frac{2}{1 - e^{-y}} - 1 \tag{M.3}$$

$$\sigma(y) = y \tag{M.4}$$

$$\sigma(y) = \begin{cases} y & y >= 0 \\ 0 & y < 0 \end{cases} \tag{M.5}$$

The exponential one is normalized and offset from zero so it ranges from -1 to 1. The last one, which simply passes through the value of y, is called the *linear* activation

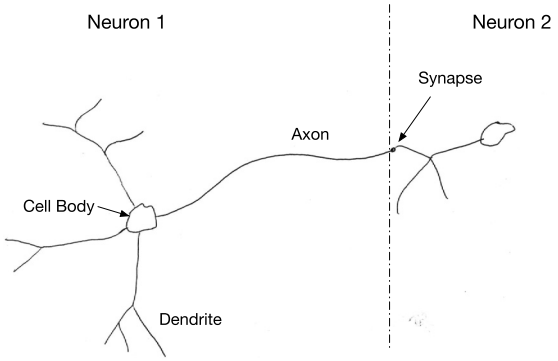


Figure M.5 A neuron connected to a second neuron. A real neuron can have 10 000 inputs!

function. The following code in the script OneNeuron.m computes and plots four activation functions for an input q. Fig. M.6 shows the four activation functions on one plot.

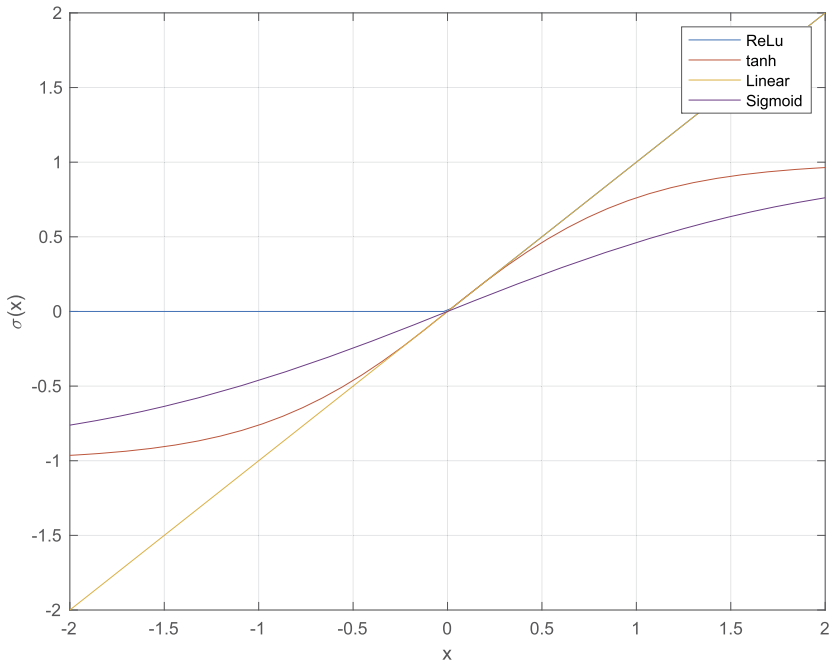


Figure M.6 Four activation functions.

Activation functions that saturate, that is reach a value of input after which the output is constant or changes very slowly, model a biological neuron that has a maximum firing

rate. These particular functions also have good numerical properties that are helpful in learning.

Let us look at a single-input neural net shown in Fig. M.7. This neuron is

$$z = \sigma(2x + 3) \tag{M.6}$$

where the weight w on the single input x is 2 and the bias b is 3. If the activation function is linear, the neuron is just a linear function of x,

$$z = y = 2x + 3 \tag{M.7}$$

Neural nets do make use of linear activation functions, often in the output layer. It is the nonlinear activation functions that give neural nets their unique capabilities.

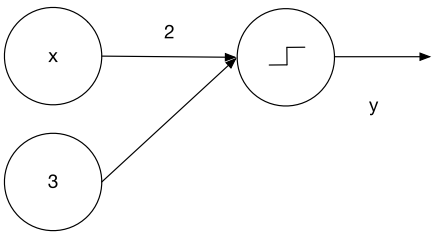


Figure M.7 A one-input neural net. The weight w is 2 and the bias b is 3. An artificial neuron consists of, at a minimum, the weight, and activation function. The bias is optional.

Let us look at the output with the above activation functions and also compare it with the threshold function The results are shown in Fig. M.8.

The tanh and exp are very similar. They put bounds on the output. Within the range $-3 \leq x < 1$ they return the function of the input. Outside those bounds they return the sign of the input, that is, they saturate. The threshold function returns zero if the value is less than 0 and 1 if it is greater than -1.5. The threshold is saying the output is only important, thus *activated*, if the input exceeds a given value. The other nonlinear activation functions say that we care about the value of the linear equation only within the given bounds. The nonlinear functions (but not steps) make it easier for the learning algorithms since the functions have derivatives. The binary step has a discontinuity at an input of zero so that its derivative is infinite at that point. Aside from the linear function (which is usually used on the output neurons), the neurons are just telling us that the sign of the linear equation is all we care about. The activation function is what makes this operation a neuron.

The XOR problem impeded the development of neural networks for some time before "deep learning", which is just a neural net with more than one layer of neurons, was developed. When the XOR problem was first studied it was shown that a single-layer neural network could not produce all four outputs from the two inputs. Look at

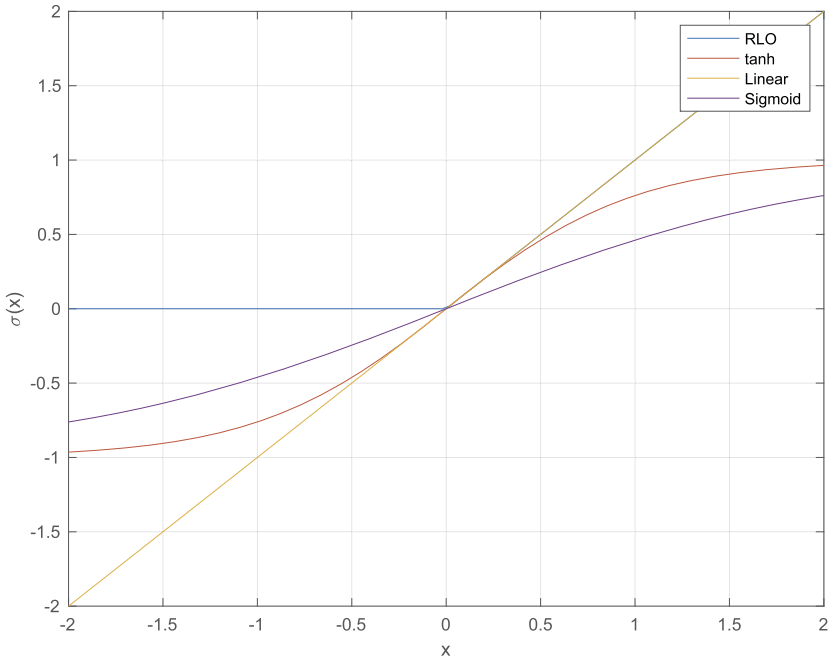


Figure M.8 The "linear" neuron compared to other activation functions from LinearNeuron.

Fig. M.9. The table on the left gives all possible inputs A and B and the desired outputs C. "Exclusive-Or" just means that if the inputs A and B are different, the output C is 1. The figure shows a single-layer network and a multilayer network, as in Fig. M.9, but with the weights labeled as they will be in the code. You can implement this easily, in just seven lines of code

```
a = 1;
b = 0;
if( a == b )
  c = 1
else
  c = 0
end
```

This type of logic was embodied in medium-scale integrated circuits in the early days of digital systems and vacuum tube-based computers before semiconductor-based computers. Try as you might, you cannot pick two weights and a bias on the single-layer network to reproduce the XOR. Marvin Minsky wrote proof that it was impossible.

A deep neural net, can reproduce the XOR. We will implement and train this network. We will explicitly write out the backpropagation algorithm that trains the

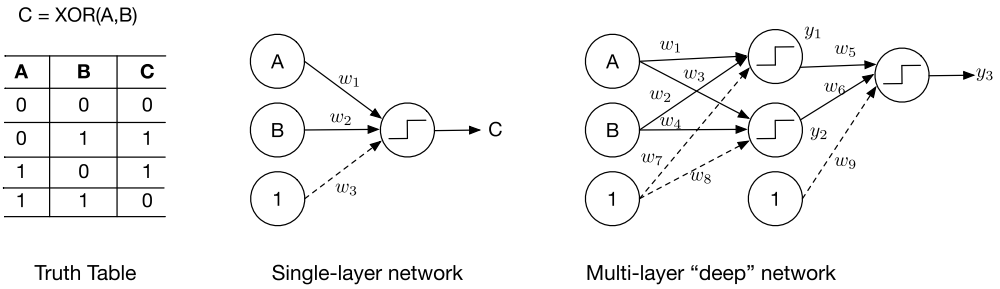


Figure M.9 Exclusive-Or (XOR) truth table and possible solution networks.

neural net from the four training sets given in Fig. M.9, i.e., (0,0), (1,0), (0,1), (1,1). We will use the tanh as the activation function in this example. tanh is a smooth function that can be differentiated, leading to an analytic result.

There are three neurons. The activation function for the hidden layer is the aforementioned hyperbolic tangent. The activation function for the output layer is linear.

$$y_1 = \tanh(w_1 a + w_2 b + w_7) \tag{M.8}$$

$$y_2 = \tanh(w_3 a + w_4 b + w_8) \tag{M.9}$$

$$y_3 = w_5 y_1 + w_6 y_2 + w_9 \tag{M.10}$$

Now, we will derive the backpropagation routine that is the learning method. The hyperbolic activation function is,

$$f(z) = \tanh(z) \tag{M.11}$$

Its derivative is

$$\frac{df(z)}{dz} = 1 - f^2(z) \tag{M.12}$$

In this derivation, we are going to use the chain rule. Assume that F is a function of y and y is a function of x. Then,

$$\frac{dF(y(x))}{dx} = \frac{dF}{dy}\frac{dy}{dx} \tag{M.13}$$

The error, at the output of the neural network, is the square of the difference between the desired output and the output. This is known as a quadratic error. It is easy to use because the derivative is simple and the error is always positive, making the lowest error the one closest to zero. Other error measures could be used.

$$E = \frac{1}{2}(c - y_3)^2 \tag{M.14}$$

The derivative of the error for w_j for the output node is

$$\frac{\partial E}{\partial w_j} = \left(y_3 - c\right) \frac{\partial y_3}{\partial w_j} \tag{M.15}$$

For the hidden nodes, it is

$$\frac{\partial E}{\partial w_j} = \psi_3 \frac{\partial n_3}{\partial w_j} \tag{M.16}$$

Expanding for all the weights,

$$\frac{\partial E}{\partial w_1} = \psi_3 \psi_1 a \tag{M.17}$$
$$\frac{\partial E}{\partial w_2} = \psi_3 \psi_1 b \tag{M.18}$$
$$\frac{\partial E}{\partial w_3} = \psi_3 \psi_2 a \tag{M.19}$$
$$\frac{\partial E}{\partial w_4} = \psi_3 \psi_2 b \tag{M.20}$$
$$\frac{\partial E}{\partial w_5} = \psi_3 y_1 \tag{M.21}$$
$$\frac{\partial E}{\partial w_6} = \psi_3 y_2 \tag{M.22}$$
$$\frac{\partial E}{\partial w_7} = \psi_3 \psi_1 \tag{M.23}$$
$$\frac{\partial E}{\partial w_8} = \psi_3 \psi_2 \tag{M.24}$$
$$\frac{\partial E}{\partial w_9} = \psi_3 \tag{M.25}$$

where

$$\psi_1 = 1 - f^2(n_1) \tag{M.26}$$
$$\psi_2 = 1 - f^2(n_2) \tag{M.27}$$
$$\psi_3 = y_3 - c \tag{M.28}$$
$$n_1 = w_1 a + w_2 b + w_7 \tag{M.29}$$
$$n_2 = w_3 a + w_4 b + w_8 \tag{M.30}$$
$$n_3 = w_5 y_1 + w_6 y_2 + w_9 \tag{M.31}$$

You can see from the derivation how this could be made recursive and applied to any number of outputs or layers. The eight adjustments at each step will be

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} \tag{M.32}$$

where η is the update gain. It should be a small number. We only have four sets of inputs. We will apply them multiple times to get the XOR weights.

The backpropagation trainer needs to find the nine elements of w. The first step is to try the network with random weights and biases. As expected, the results of the neural network with random weights and biases are not good and it does not reproduce the XOR. After training, the neural network reproduces the XOR problem very well, as shown below. If you change the initial weights and biases, you may find that you get bad results. This is because the simple gradient method implemented here can fall into local minima from which it cannot escape. This is an important point about finding the best answer. There may be many good answers, which are local optimums, but there will be only one best answer. There is a vast body of research on how to guarantee that a solution is globally optimal. Other training systems, like simulated annealing and genetic algorithms, are less likely to become stuck in a local optimal.

Example M.1 shows the weights and biases converging and also shows the mean output error over all four inputs in the truth table going to zero. If you try other starting weights and biases this may not be the case. Other solution methods, such as Genetic Algorithms [5], Electromagnetism based [6], and Simulated Annealing [7] are less susceptible to falling into local minima but can be slow. A good overview of optimization specifically for machine learning is given by Bottou [8].

You might wonder how this compares to a set of linear equations. If we remove the activation functions we get

$$y_3 = w_9 + w_6 w_8 + w_5 w_7 + a(w_1 w_5 + w_3 w_6) + b(w_2 w_5 + 2_r w_6) \tag{M.33}$$

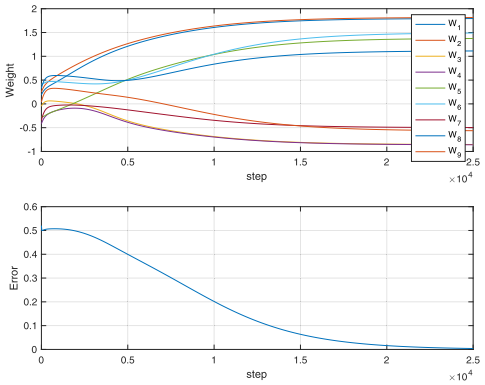
This reduces to just three independent coefficients

$$y_3 = k_1 + k_2 a + k_3 b \tag{M.34}$$

One is a constant, and the other two multiply the inputs. Writing the four possible cases in matrix notation we get

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = k_1 + \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} k_2 \\ k_3 \end{bmatrix} \tag{M.35}$$

```
% Training data - also the truth data
a        = [0 1 0 1];
b        = [0 0 1 1];
c        = [0 1 1 0];

% First try implementing random weights
w0       = [ 0.1892; 0.2482; -0.0495; -0.4162;
            -0.2710;...
             0.4133; -0.3476; 0.3258; 0.0383];
cR       = XOR(a,b,w0);

fprintf('\nRandom Weights\n')
fprintf('    a      b      c\n');
for k = 1:4
  fprintf('%5.0f %5.0f %5.2f\n',a(k),b(k),cR(k));
end

% Now execute the training
w        = XORTraining(a,b,c,w0,25000,0.001);
cT       = XOR(a,b,w);

fprintf('\nWeights and Biases\n')
fprintf('   Initial    Final\n')

for k = 1:9
  fprintf('%d  %7.4f %7.4f\n',k,w0(k),w(k));
end

fprintf('\nTrained\n')
fprintf('    a      b      c\n');
for k = 1:4
  fprintf('%5.0f %5.0f %5.2f\n',a(k),b(k),cT(k));
end
```




```
Random Weights
    a      b      c
    0      0   0.26
    1      0   0.19
    0      1   0.03
    1      1  -0.04

Weights and Biases
   Initial    Final
1   0.1892   1.7933
2   0.2482   1.8155
3  -0.0495  -0.8535
4  -0.4162  -0.8591
5  -0.2710   1.3744
6   0.4133   1.4893
7  -0.3476  -0.4974
8   0.3258   1.1124
9   0.0383  -0.5634

Trained
    a      b      c
    0      0   0.00
    1      0   1.00
    0      1   1.00
    1      1   0.01
```

Example M.1: XOR neural-network weights converge during training.

We can get close to a working XOR if we choose

$$\begin{bmatrix} k_1 \\ k_2 \\ k_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \tag{M.36}$$

This makes three out of four equations correct. There is no way to make all four correct with just three coefficients, as Minsky proved. The activation functions separate the coefficients and allow us to reproduce the XOR. This is not surprising because the XOR is not a linear problem.

M.7. Static Earth sensors

Earth sensors were some of the earliest spacecraft sensors. They are still very popular for CubeSats. Many early satellites were for communications or Earth observations so during most of the satellite life it stared at the Earth, which means zeroing the angles between the axis pointing at the Earth and the nadir vector. In many cases, the yaw error, the error around the nadir vector or vector to the Earth, was not important or could be controlled through the momentum of the satellite if it was a momentum-bias or dual-spin spacecraft. Earth sensors were either static, without moving parts, or scanning. Scanning was done by rotating the satellite, rotating a mirror using a motor, or vibrating a mirror with a torsion bar. Most sensors operated in the infrared wavelengths because the radiation at those wavelengths is more uniform than that in the visible bands and is unaffected by eclipses.

Fig. M.10 shows the attitude (also known as orientation) geometry for small angles from ideal Earth pointing. Roll is about the x-axis and pitch is about the y-axis.

Fig. M.11 shows how a static Earth sensor operates. The image on the left shows the Earth-sensor elements, represented by the circles centered on the Earth. The roll and pitch angles are zero. Each element sees part of the Earth and part of space so they all produce the same signal. The picture on the right shows the sensor with a roll and pitch angle. Some elements see just space, some see just the Earth, and some both the Earth and space but their output will be different from the zero angle cases. By comparing the outputs of the sensors we get a measurement of roll and pitch. The sensors see light in the CO_2 band, around 14 micrometers wavelength, so the Earth looks uniform.

Static Earth sensors from the 1960s had built-in logic to go from sensor outputs to roll and pitch. Given the hardware limitations, the algorithms were quite simple and could be implemented with a few transistors or medium-scale integrated circuits. In this section, we show how a neural network can be trained to take the sensor inputs and return roll-and-pitch measurements.

Example M.2 shows the neural-network training and execution.

The first part of the script generates the training data. It calls ISSOrbit to get the Keplerian orbital elements for the International Space Station (ISS). It then gets the default data structure from StaticEarthSensor. The script then converts orbital elements into position and velocity vectors using RVOrbGen and computes the quaternion from the Earth-centered inertial frame (ECI) to the local vertical/local lorizontal (LVLH), which is the normal Earth-pointing orientation. We then create a four-element Earth-sensor model, replacing the default fields in d. This line of code draws the sensor.

The top plot shows the performance over the range of input roll-and-pitch angles. The middle plot shows the training data. The last plot shows roll-and-pitch measurement over the test orbit when the true roll is two degrees and the true pitch is zero.

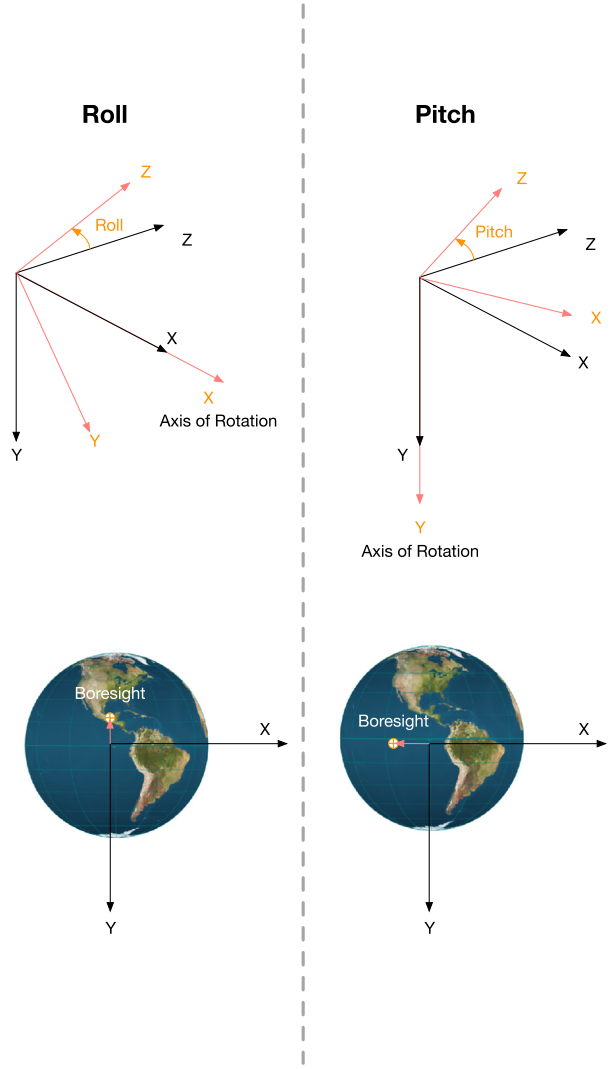
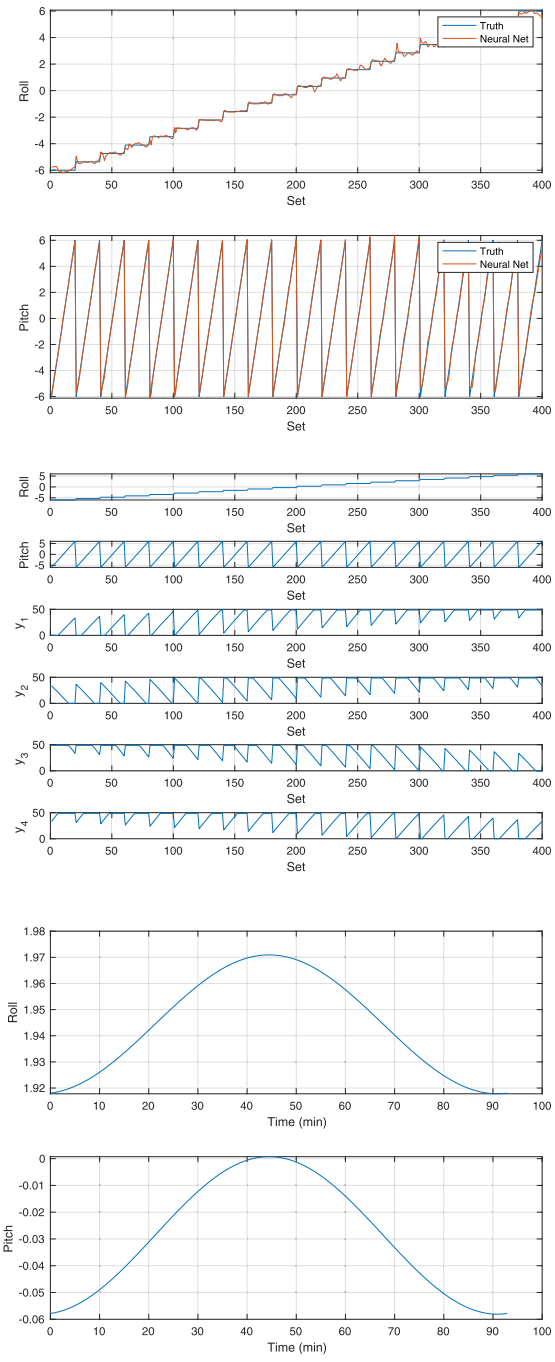


Figure M.10 Attitude geometry showing roll and pitch.

Since this neural network was only trained at one point in the orbit, that is at one altitude, it will not match roll and pitch exactly over the entire orbit.

Fig. M.12 shows the Earth sensor and Fig. M.13 the training GUI from MATLAB®. The GUI shows the structure of the neural network. Fig. M.12 shows the Earth sensor and the training GUI from MATLAB. The GUI shows the structure of the neural network.

```
degToRad     = pi/180;
rE           = 6378.165;
[el,jD0]     = ISSOrbit;
d            = StaticEarthSensor;
[r,v,t]      = RVOrbGen(el);
rMean        = mean(Mag(r));
qECIToLVLH   = QLVLH(r,v);
d.el         = 64*ones(1,4)*degToRad;
d.az         = [0 pi/2 pi 3*pi/2] + pi/4;
d.pAng       = 4*[  1    1   -1   -1
                    1   -1   -1    1];

n            = 20;
roll         = linspace(-6,6,n);
pitch        = linspace(-6,6,n);
i            = 1;
y            = zeros(4,n*n);
x            = zeros(2,n*n);

StaticEarthSensor(qECIToLVLH(:,1),r(:,1),d)

for j = 1:n
  for k = 1:n
    rJ           = roll(j);
    pK           = pitch(k);
    mRoll        = [1 0 0;0 CosD(rJ) -SinD(rJ);0
          SinD(rJ) CosD(rJ)];
    mPitch       = [CosD(pK) 0 -SinD(pK);0 1 0;
          SinD(pK) 0 CosD(pK)];
    qLVLHToBody = Mat2Q(mRoll*mPitch);
    qECIToBody   = QMult(qECIToLVLH(:,1),
          qLVLHToBody);

    y(:,i)       = StaticEarthSensor(qECIToBody,r
          (:,1),d);
    x(:,i)       = [roll(j);pitch(k)];
    i            = i + 1;
  end
end

% Neural net training
net          = feedforwardnet(20); % Generate the
      neural net structure

net          = configure( net, y, x ); % Configure
      based on inputs and outputs
net.layers{1}.transferFcn = 'poslin'; % Set as
      purelin
net.name     = 'Earth Sensor';
net          = train(net,y,x); % Train the network
c            = sim(net,y); % Simulate the neural
      network
leg          = {'Truth' 'Neural Net'};

PlotSet(1:size(c,2),[x;c],'x label','Set','y
      label',{'Roll' 'Pitch'},...
       'figure title','Neural Network','plot set
            ',{[1 3],[2 4]},...
       'legend',{leg leg});

yL = {'Roll' 'Pitch' 'y_1' 'y_2' 'y_3' 'y_4'};
PlotSet(1:size(c,2),[x;y],'x label','Set','y
      label',yL,'Neural Network Data')

%% Testing
n     = length(t);
roll  = 2;
pitch = 0;
c     = zeros(2,n);
for k = 1:n
  rJ           = roll;
  pK           = pitch;
  mRoll        = [1 0 0;0 CosD(rJ) -SinD(rJ);0
        SinD(rJ) CosD(rJ)];
  mPitch       = [CosD(pK) 0 -SinD(pK);0 1 0;SinD(
        pK) 0 CosD(pK)];
  qLVLHToBody = Mat2Q(mRoll*mPitch);
  qECIToBody   = QMult(qECIToLVLH(:,k),qLVLHToBody
        );
  y            = StaticEarthSensor(qECIToBody,r(:,
        k),d);
  c(:,k)       = sim(net,y');
end

[t,tL]   = TimeLabl(t);
s        = sprintf('Roll = %8.2f deg Pitch = %8.2f
      deg',roll,pitch);

PlotSet(t,c,'x label',tL,'y label',{'Roll' '
      Pitch'},'figure title',s)
```



Example M.2: Neural network trained to produce pitch-and-roll measurements from sensor measurements.

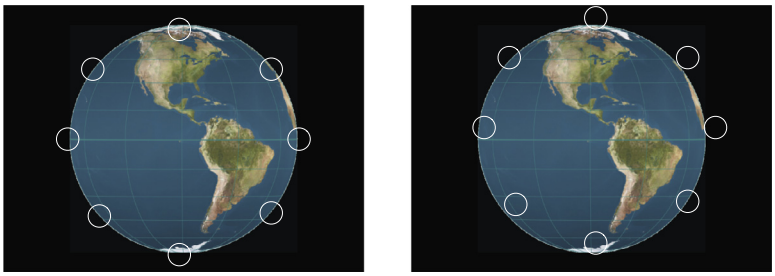


Figure M.11 Static Earth-sensor operation. The circles represent the sensor elements. The left shows the geometry with zero roll and pitch. The right shows the geometry with roll-and-pitch angles.

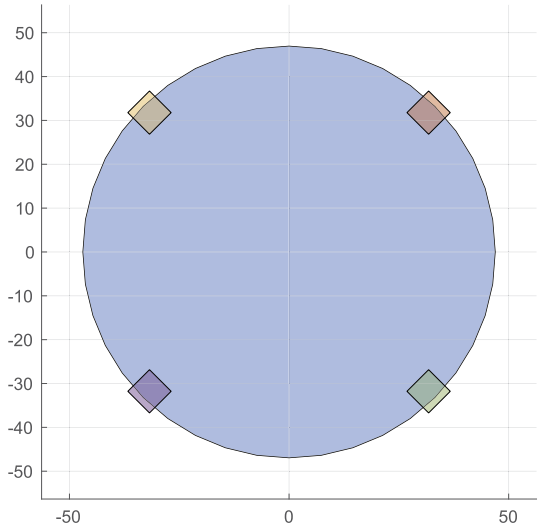


Figure M.12 The Earth sensor.

M.8. Expert systems

Expert systems are a way of embodying knowledge about a system in a database and a means for making decisions based on that data. Traditionally, this has been done by using logical statements in computer code. For example, a spacecraft might check an electrical-current measurement for a reaction wheel to see if it is drawing power.

```
{
        if( measuredCurrent > 0 )
         % Use the reaction wheel
        else
         % Use a backup wheel
```

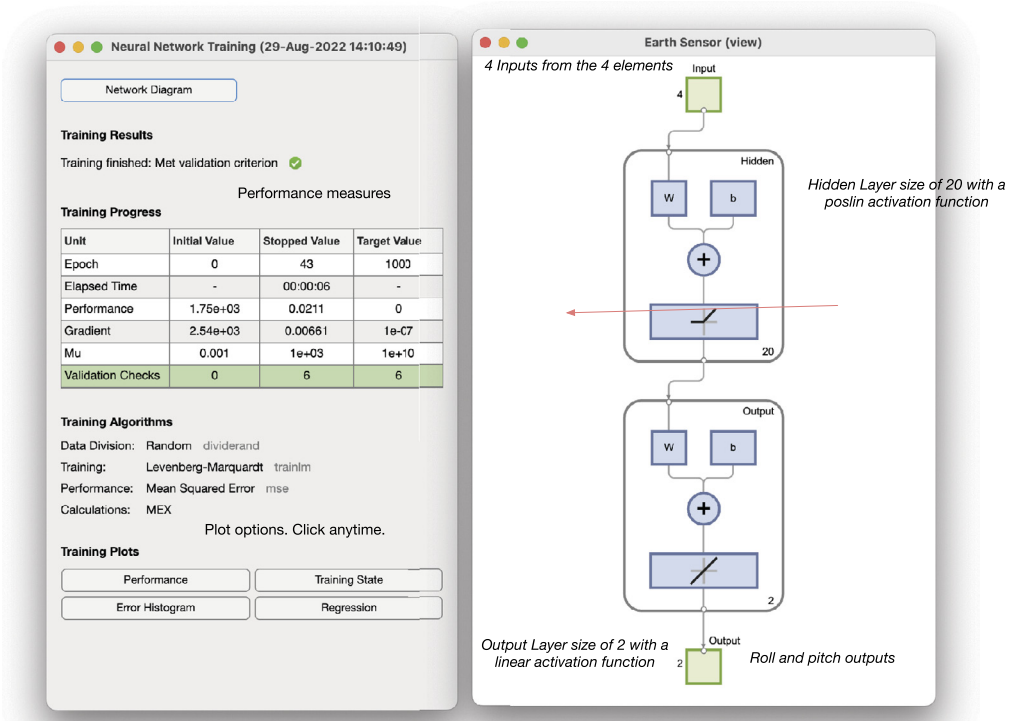


Figure M.13 The training GUI.

```
        end
}
```

This can work well for small numbers of these constructs. For very large numbers of logical statements, the software becomes very difficult to maintain and test. Expert systems are an elegant way to provide such capabilities.

There are two broad classes of expert systems, rule-based and case-based. Rule-based systems have a set of rules that are applied to make a decision. These systems provide a way of automating the process when decision-making involves hundreds or thousands of rules. Case-based systems decide by example, that is, a set of predefined cases. NASA CLIPS (C Language Integrated Production System) is a rule-based language that was developed by NASA's Johnson Space Center. NASA Deep Space 1 [9] used Remote Agents (RA) that employed a rule-based expert system for mission planning and execution and fault detection.

Learning in the context of an expert system depends strongly on the configuration of the expert system. There are three primary methods, which vary in the level of

autonomy of learning and the average generalization of the new knowledge of the system.

The least autonomous method of learning is the introduction of new rule sets in simple rule-based expert systems. Learning of this sort can be highly tailored and focused, but is done entirely at the behest of external teachers. In general, very specific rule-based systems with extremely general rules tend to have issues with edge cases that require exceptions to their rules. Thus, this type of learning, while easy to manage and implement, is neither autonomous nor generalizable.

The second method is fact gathering. The expert system makes decisions based on the known cause-and-effect relationships along with an evolving model of the world; learning then is broken up into two subpieces. Learning new cause-and-effect system rules is very similar to the type of learning described above, requiring external instruction, but can be more generalizable (as it is combined with more general world knowledge than a simple rule-based system might have). Learning new facts, however, can be autonomous and involves the refinement of the expert system's model of reality by increasing the amount of information that can be taken advantage of by automated reasoning systems.

The third method is fully autonomous-based reasoning, where actions and their consequences are observed, leading to inferences about what prior and action combinations lead to what results. For instance, if two similar actions result in positive results, then those priors that are the same in both cases can begin to be inferred as necessary preconditions for a positive result from that action. As additional actions are seen, these inferences can be refined and confidence can increase in the predictions made.

The three methods are listed in increasing difficulty of implementation. Adding rules to a rule-based expert system is quite straightforward, though rule dependencies and priorities can become complicated. Fact-based knowledge expansion in automated reasoning systems is also fairly straightforward, once suitably generic sensing systems for handling incoming data are set up. The third method is by far the most difficult; however, rule-based systems can incorporate this type of learning. In addition, more general pattern-recognition algorithms can be applied to training data (including online, unsupervised training data) to perform this function, learning to recognize, e.g., with a neural network, patterns of conditions that would lead to positive or negative results from a given candidate action. The system can then check possible actions against these learned classification systems to gauge the potential outcome of the candidate's actions.

The following example explores case-based reasoning systems. This is a collection of cases with their states and values given by strings. The code in this chapter can be made to learn by feeding back the results of new cases into the case-based system.

The knowledge base consists of states, values, and production rules. There are four parts to a new case: the case name, the states, values, and the outcome. A state can have multiple values.

The state catalog is a list of all of the information that will be available to the reasoning system. It is formatted as states and state values.

The default catalog is shown below for a reaction-wheel control system. A MATLAB cell array of acceptable or possible values for each state follows the state definition

```
{
        {'wheel-turning'},      {'yes','no'};
        {'power'},              {'on','off'};
        {'torque-command'},     {'yes','no'}
}
```

The database of cases is designed to detect failures. There are three things to check to see if the wheel is working. If the wheel is turning and power is on and there is a torque command then it is working. The wheel can be rotating without a torque command or with the power off because it would just be spinning down from prior commands. If the wheel is not turning the possibilities are that there is no torque command or the power is off.

Once cases are defined from the catalog, the system can be tested The function CBREngine implements the case-based reasoning engine. The algorithm finds the total fraction of the cases that match to determine if the example matches the stored cases. The engine is matching values for states in the new case against values for states in the case database. It weights the results by the number of states. If the new case has more states than an existing case, the result is the number of states in the database case divided by the number of states in the new case. If more than one case matches the new case, and the outcomes for the matching cases are different, the outcome is declared "ambiguous". If they are the same, it gives the new case that outcome. The case names make it easier to understand the results.

The demo script, Example M.3 builds the system and runs some cases. The script matches two cases but because their outcome is the same, the wheel is declared working. For the wheel power, the power could be on or off, hence it is declared ambiguous.

This example is for a very small case-based expert system with a binary outcome. Multiple outcomes can be handled without any changes to the code. However, the matching process is slow as it cycles through all the cases.

M.9. Reinforcement learning

M.9.1 Introduction

Reinforcement learning trains an agent to perform a task within a specified environment. The architecture of an agent is shown in Fig. M.14. The policy is the mechanism that turns observations, sensor measurements for example, into actions, such as control torques. A control system is an agent that is trained by the control-system designer,

```
system = BuildExpertSystem( [], 'id',1,...
'catalog state name','wheel-turning',...
'catalog value',{'yes','no'},...
'id',2,...
'catalog state name','power',...
'catalog value',{'on' 'off'},...
'id',3,...
'catalog state name','torque-command',...
'catalog value',{'yes','no'},...
'id',1,...
'case name', 'Wheel operating',...
'case states',{'wheel-turning', 'power', 'torque-
    command'},...
'case values',{'qyes' 'on' 'yes'},...
'case outcome','working',...
'id',2,...
'case name', 'Wheel power ambiguous',...
'case states',{'wheel-turning', 'power', 'torque-
    command'},...
'case values',{'yes' {'on' 'off'} 'no'},...
'case outcome','working',...
'id',3,...
'case name', 'Wheel broken',...
'case states',{'wheel-turning', 'power', 'torque-
    command'},...
'case values',{'no' 'on' 'yes'},...
'case outcome','broken',...
'id',4,...
'case name', 'Wheel turning',...
'case states',{'wheel-turning', 'power'},...
'case values',{'yes' 'on'},...
'case outcome','working',...
'match percent',80);

newCase.state  = {'wheel-turning', 'power', '
    torque-command'};
newCase.values = {'yes','on','no'};
newCase.outcome = '';

[newCase.outcome, pMatch] = CBREngine( newCase,
    system );

fprintf(1,'New case outcome: %s\n\n',newCase.
    outcome);

fprintf(1,'Case ID Name                                                    Percentage Match\n');
for k = 1:length(pMatch)
  fprintf(1,'Case %d: %-30s %4.0f\n',k,system.
      case(k).name,pMatch(k)*100);
end
```

```
New case outcome: working

Case ID Name                                                    Percentage Match
Case  1: Wheel working                                          67
Case  2: Wheel power ambiguous                                  67
Case  3: Wheel broken                                           33
Case  4: Wheel turning                                          44
```

Example M.3: Case-based expert system for reaction-wheel fault detection.

usually through the computation of a set of parameters with a fixed structure, such as a PID controller. A PID that learned its gains through repeated tests would fit into the architecture shown in Fig. M.14. A reinforcement-learning system can go beyond this by creating the algorithm itself. The policy is often a multilayer, or deep, neural network. Through training, the neuron weights and biases are chosen to produce a policy that maximizes the reward. The reward is based on how well the agent achieves the objective. For example, an agent would get a big reward if it maintains zero pointing errors.

Agents can be categorized into three groups:

- Discrete actions and discrete observations;

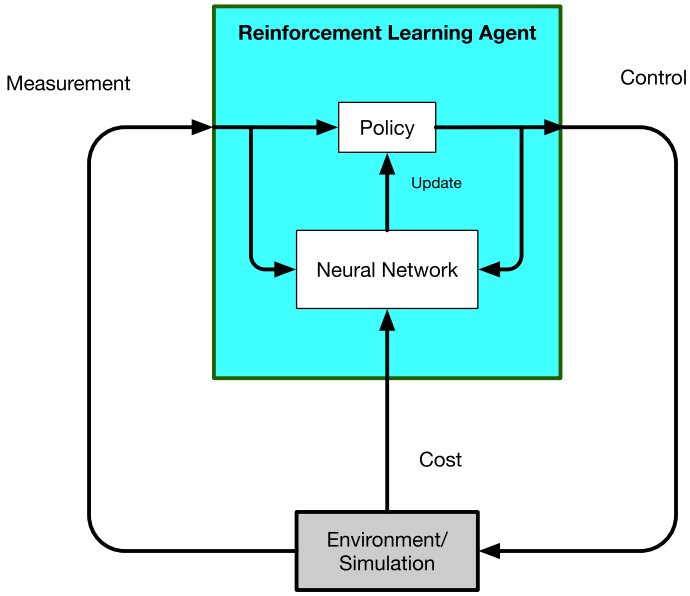


Figure M.14 Reinforcement-learning architecture.

- Discrete actions and continuous observations;
- Both continuous actions and observations.

A bang-zero-bang controller, such as the one implemented in the phase-plane logic on the Space Shuttle Orbiter, is an example of discrete actions and continuous observations. Continuous actions and observations would describe any conventional control system.

The maneuver algorithm will use a Deep Deterministic Policy Gradient (DDPG) algorithm [10]. It is a model-free, online, offpolicy reinforcement-learning method. It works with continuous observations and actions. The policy maps observations to actions. The policy is trained using data from simulations. The policy is updated with the latest observations. Offpolicy agents create a buffer of observations and use that to update the policy. It can always use old observation/action pairs. A DDPG agent is an actor–critic reinforcement-learning agent that searches for an optimal policy.

Within a DDPG agent, there are actors and critics. An actor performs the action, in this case, it commands the reaction-wheel torque, based on the observations. A critic evaluates the performance. Both the actor and critic are neural networks.

Reinforcement learning is based on maximizing a reward, which is no different from minimizing a cost in an optimal control problem. For a spacecraft maneuver the reward is based on the following:

1. Reaching the desired state;
2. The time it takes to reach the final state;

3. The amount of momentum the wheels need to store.

The desired state includes both the desired quaternion and the desired angular rate. During the maneuver, only the third criterion can be evaluated. The second criteria are evaluated once the first criteria are achieved.

M.9.2 Optimal attitude trajectory

This approach is similar to constrained optimization using a direct optimization method. The constraint is the final state, which must be the target. It is usually easiest to specify a fixed terminal time although time can be a value that the algorithm finds. The trajectory is broken up into segments with constant control on each segment. The optimizer then adjusts the controls until the final state is met. If the optimizer has difficulty, it can be given more time to find the solution. The cost is the sum of the magnitudes of the applied torque. The control is also constrained. The optimization needs a reasonable guess of the control to start. In this case, a bang-bang maneuver for a single axis is a reasonable first guess. The torque is a positive constant and then a negative constant. The control is bounded by the maximum-allowable torque.

Example M.4 shows the optimization code. The terminal state is the constraint. The software uses a fixed end time. The body rates must be zero and the quaternion must match the target quaternion at the terminal condition. The constraint is a six-vector that is the desired terminal attitude and angular rates. The attitude part is computed from the last three elements of the delta quaternion computed by

$$q_\delta = q^T \otimes q_T \tag{M.37}$$

where q is the terminal quaternion at the selected end time. If the delta quaternion is

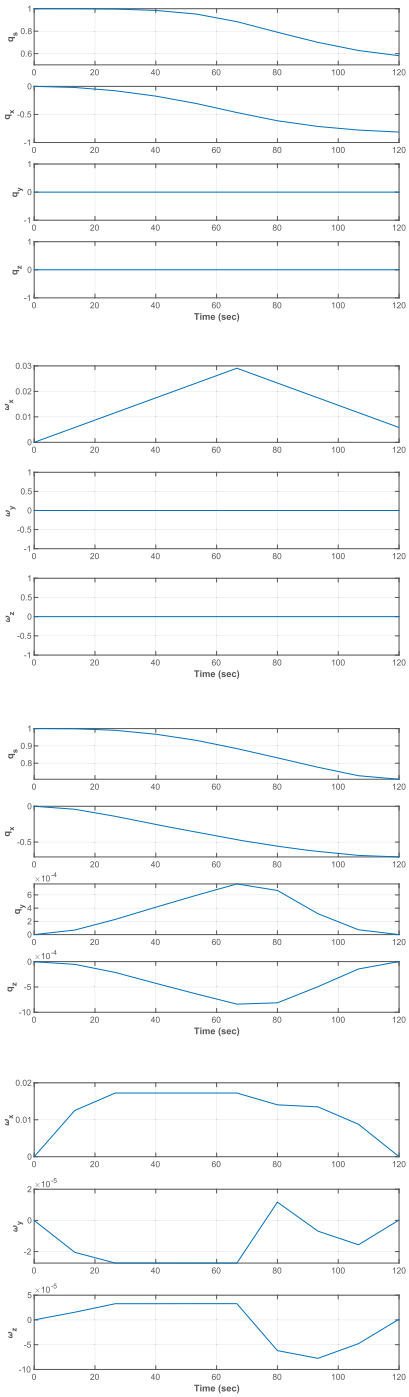
$$q_\delta = \begin{bmatrix} s \\ v \end{bmatrix} \tag{M.38}$$

the constraint equation is

$$c = \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{M.39}$$

Ideally, this is a zero vector. Using the entire quaternion leads to singularities when the Hessian matrix is computed. fmincon is used with the interior-point method. This method computes a numerical Hessian matrix. The Hessian provides information on the curvature of the problem thus speeding convergence. Interior-point methods traverse the feasible region's interior, not the boundary. Only ten segments are used. That is, the control only changes ten times throughout the optimization. Larger numbers of points slow the process and may not get better results. In this case, the problem has a known solution with just two segments.

```
% Number of control segments
n               = 10;
torqueMax       = 0.01;
d.dRHS          = RHSGyrostat; % The data structure
d.n             = n; % Number of decision variable
      increments
d.tEnd          = 2*60;
d.t             = linspace(0,d.tEnd,n);
theta           = pi/2;
qEnd            = AU2Q(theta,[1;0;0]);
d.xEnd          = [qEnd;zeros(6,1)];

% Initial state
x               = [1;zeros(9,1)];
d.x             = x;

% fmincon options
opts = optimset(  'Display','iter-detailed',...
 'TolFun',1e-4,...
 'algorithm','interior-point',...
 'TolCon',1e-4,...
 'MaxFunEvals',15000);

% The cost is the sum of the torque magnitudes
costFun  = @(x) MnvrCost(x,d);

% The numerical integration of the state is in
      the constraint function
constFun = @(x) MnvrConst(x,d);

% x-axis maneuver
torque = theta*d.dRHS.inr(1,1)/(d.tEnd/2)^2;
oN = ones(1,n/2);
u0 = [torque*[-oN,oN];zeros(2,n)];

% Lower and upper bounds for the control
lB = -torqueMax*ones(3,n);
uB =  torqueMax*ones(3,n);

% Find the optimal decision variable
u = fmincon(costFun,u0,[],[],[],[],lB,uB,constFun
      ,opts);

costBase    = MnvrCost(u0,d)
costOptimal = MnvrCost(u,d)

%% Run the simulations
SimulationRWA( x, d.t, d.dRHS, u0 );
SimulationRWA( x, d.t, d.dRHS, u );


%% Maneuver cost
function cost = MnvrCost( u, d )

magU = Mag(u);
cost = sum(magU);

end

%% Maneuver nonlinear constraint
function [cIn, cEq] = MnvrConst( u, d )

xP  = SimulationRWA( d.x, d.t, d.dRHS, u );

% We don't have any nonlinear inequality
      contraints
cIn = [];

cQ  = QMult(QPose(xP(1:4,end)),d.xEnd(1:4));

cEq = [abs(cQ(2:4));xP(5:7,end)];

end
```



Example M.4: Optimal attitude trajectory.

The optimizer iterations are shown below. The first column is the number of iterations. The second, f(x) is the cost, in this case, the sum of the torque magnitudes. The second column is the total function count. The third column, Feasibility, is how well it is matching the landing constraint of zero quaternion error and angular rates. Ideally, it is zero when the constraint is matched. The First-Order Optimality is how close the solution is to the optimal solution. The Norm step size is the Norm (essentially magnitude) of the control step it is taking. It uses only about 1/3 of the maximum allowable steps.

```
                                                First-order      Norm of
 Iter F-count            f(x)  Feasibility      optimality          step
    0       31    4.363323e-03    1.641e-01    1.044e-02
    1       62    3.979051e-03    1.514e-03    1.069e+01    1.693e-03
    2       93    3.043139e-03    2.629e-06    3.144e+00    6.527e-04
    3      165    3.043136e-03    1.147e-06    1.073e-02    4.323e-09
    4      196    2.796689e-03    2.095e-01    3.722e+00    7.572e-04
    5      229    2.756049e-03    1.569e-01    3.274e+00    2.598e-04
    6      263    2.488857e-03    1.372e-01    2.905e+00    4.419e-04
    7      300    2.715273e-03    8.616e-04    1.089e-02    2.207e-04
    8      340    2.706251e-03    1.366e-06    5.009e-01    1.217e-04
    9      379    2.678182e-03    8.713e-03    5.014e-01    3.089e-05
   10      411    2.657084e-03    8.690e-03    1.498e+00    3.041e-05

  140     4990    2.586397e-03    4.436e-06    7.234e-01    4.630e-09
  141     5022    2.586396e-03    4.677e-06    7.753e-01    4.633e-09
  142     5057    2.586385e-03    5.219e-06    7.342e-01    9.255e-09
  143     5089    2.586384e-03    5.024e-06    7.342e-01    1.853e-08
  144     5126    2.586372e-03    8.224e-06    7.898e-01    9.244e-09
  145     5158    2.586371e-03    8.472e-06    8.136e-01    9.261e-09
  146     5193    2.586398e-03    3.027e-06    4.869e-01    1.850e-08
  147     5234    2.586396e-03    3.020e-06    4.875e-01    2.027e-09

The optimization stopped because the relative changes in all elements of the state $x$
     are
less than options.StepTolerance = 1.000000e-10, and the relative maximum constraint
violation, 3.020445e-06, is less than options.ConstraintTolerance = 1.000000e-04.

costBase =

    0.0044

costOptimal =

    0.0026
```

The list only shows the first eleven and last eight steps. The cost of the optimal maneuver is almost half that of the simple maneuver. As the plots show, the spacecraft does not do a single-axis maneuver. The rates for the other axes are very small but nonzero.

M.9.3 Single-axis optimal attitude trajectory

A single-axis problem is useful to explore reinforcement learning. The problem is to change the angle by a predetermined amount. The dynamical equations are

$$\dot{\theta} = \omega \tag{M.40}$$

$$\dot{\omega} = u \tag{M.41}$$

The DDPG agent applies reinforcement learning to this problem. The reinforcement learning system must maximize the reward that is a combination of

- The maximum torque;
- The rate error at the end;
- The angle error.

The reward algorithm is as follows.

```
% Check terminal condition
angError     = this.State(1)-this.thetaTarget;
IsDone       = abs(angError) < this.thetaError
    ;
this.IsDone = IsDone;

% When done we want the angular rate to be
    zero
if( IsDone )
  c = 100*exp(-1500*Mag(this.State(2)));
else
  c = 0;
end

% Negative reward for torque and error from
    the target
Reward = -0.001*torque^2 - 0.01*angError^2 +
    c;
```

When the rate error is zero, c is 100. At each step, the reward for being far from the target and for large torques is negative. The weights on the various components will determine the overall performance

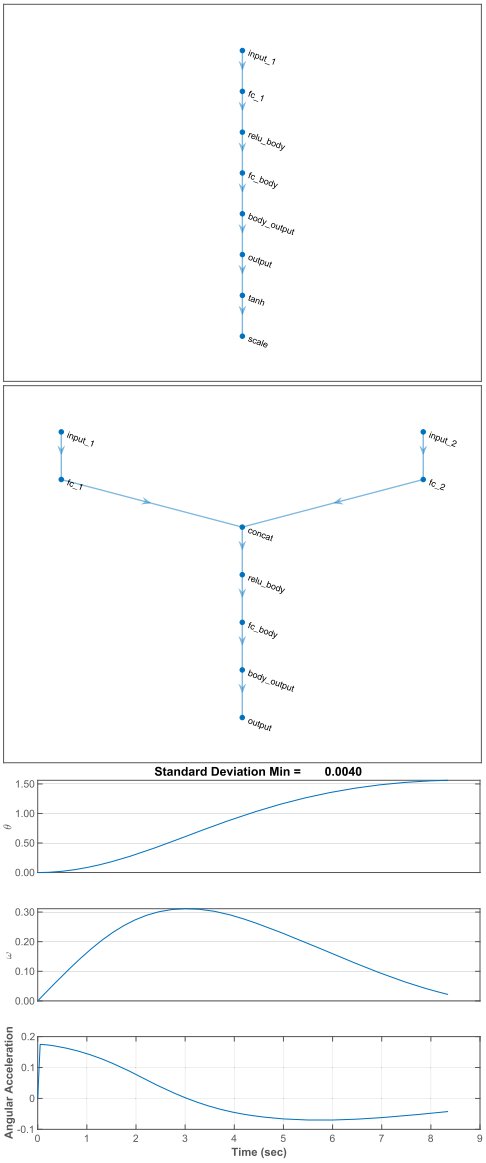
DDPG is an actor–critic approach. The actor selects an action based on a policy while the policy is evaluated by a critic using a value function [10].

The training window is shown in Fig. M.15. The dark blue line, which is the average reward since the start, should converge on the best reward.

Though it is not easy to see, close inspection shows that Q0 does converge to the mean reward.

The first diagram is a flow diagram for the actor network, the one that commands the angular acceleration. The second is the critic network that evaluates the performance. The minimum noise, and other parameters, result in a linear control law.

```
env      = OneAxisClass;

obsInfo = getObservationInfo(env);
actInfo = getActionInfo(env);
disp(obsInfo);
disp(actInfo);

initOpts   = rlAgentInitializationOptions('
        NumHiddenUnit',128);
agent      = rlDDPGAgent(obsInfo,actInfo,initOpts)
        ;

% Customize the agent
agent.AgentOptions.NoiseOptions.
        StandardDeviationDecayRate = 1e-5;
agent.AgentOptions.NoiseOptions.
        StandardDeviationMin = 0.02*agent.
        ActionInfo.UpperLimit;
agent.AgentOptions.CriticOptimizerOptions =
        rlOptimizerOptions('LearnRate',1e-4);
agent.AgentOptions.ActorOptimizerOptions =
        rlOptimizerOptions('LearnRate',1e-4);
actorNet   = getModel(getActor(agent));
criticNet = getModel(getCritic(agent));

NewFig('Actor Network')
plot(layerGraph(actorNet))
NewFig('Critic Network')
plot(layerGraph(criticNet))
disp('Critic Network')
disp(criticNet.Layers)
disp('Actor Network')
disp(actorNet.Layers)

doTraining = true;

maxsteps = 200;
maxepisodes = 1500;
trainingOpts = rlTrainingOptions(...
    'MaxEpisodes',maxepisodes,...
    'MaxStepsPerEpisode',maxsteps,...
    'Verbose',true,...
    'Plots','training-progress',...
    'StopTrainingCriteria','EpisodeReward',...
    'StopTrainingValue',1000);

if( doTraining )
    % Train the agent.
    trainingStats = train(agent,env,trainingOpts)
            ;
end

simOptions = rlSimulationOptions('MaxSteps',floor
        (1.2*trainingStats.EpisodeSteps(end)));
experience = sim(env,agent,simOptions);

t   = env.Ts*experience.Observation.
        SpacecraftStates.Time';
x   = experience.Observation.SpacecraftStates.Data
        ;
x   = reshape(x,2,size(x,3));
u   = experience.Action.torque.Data;
u   = [0 reshape(u,1,size(u,3))];
yL = {'\theta' '\omega' 'Angular Acceleration'};
s   = sprintf('Standard Deviation Min = %12.4f'
        ,...
  agent.AgentOptions.NoiseOptions.
          StandardDeviationMin);
TimeHistory(t,[x;u],yL,s);
```



Example M.5: One-axis reinforcement learning.

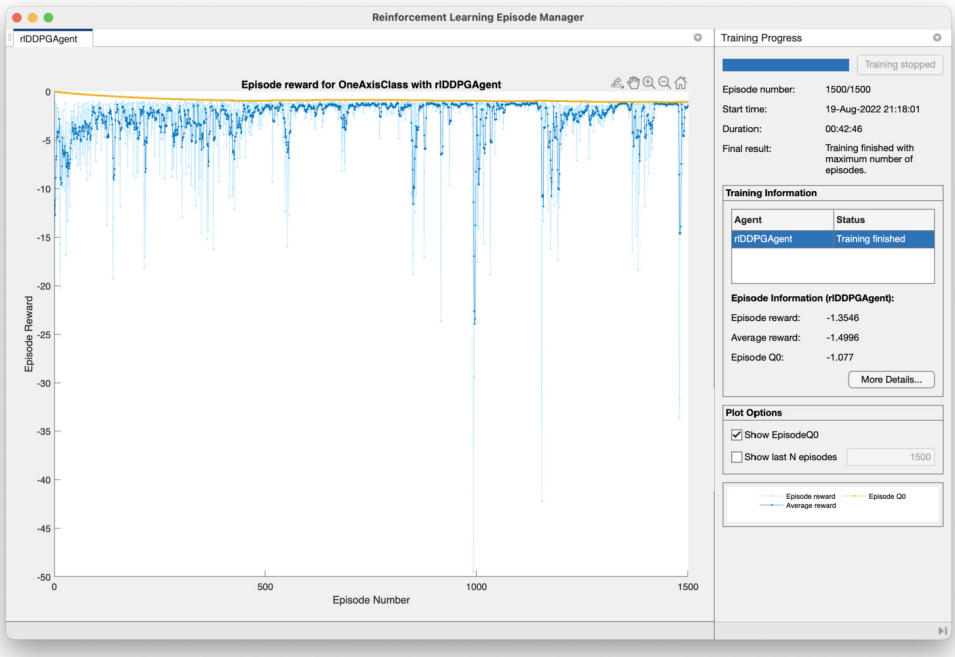


Figure M.15 Reinforcement-learning training window. The agent converges after 1500 episodes. The light blue line is the episode reward, the dark blue is the average reward since the start of training.

References

[1] G.A. Dorais, D. Kortenkamp, Deep Space One Remote Agent, https://personal.traclabs.com/~korten/tutorial/arc.pdf.

[2] D. Bernard, G. Dorais, C. Fry, E. Gamble, B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. Nayak, B. Pell, K. Rajan, N. Rouquette, B. Smith, B. Williams, Design of the Remote Agent experiment for spacecraft autonomy, in: 1998 IEEE Aerospace Conference Proceedings (Cat. No. 98TH8339), vol. 2, 1998, pp. 259–281.

[3] P. Graham, ANSI Common Lisp, Prentice-Hall, 1995.

[4] M.D. Rayman, P. Varghese, D.H. Lehman, L.L. Livesay, Results from the Deep Space 1 technology validation mission, Acta Astronautica 47 (2) (2000) 475–487, Space an Integral Part of the Information Age.

[5] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1988.

[6] I. Birbil, S.-C. Fang, An electromagnetism-like mechanism for global optimization, Journal of Global Optimization 25 (03) (2003) 263–282.

[7] L.R. Rere, M.I. Fanany, A.M. rymurthy, Simulated annealing algorithm for deep learning, Procedia Computer Science 72 (2015) 137–144.

[8] L. Bottou, F.E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Review 60 (2016) 223–311.

[9] D. Bernard, R. Doyle, J. Riedel, N. Rouquette, J. Wyatt, M. Lowry, P. Nayak, Autonomy and software technology on NASA's Deep Space One, IEEE Intelligent Systems & Their Applications 14 (06) (1999) 10–15.
[10] D.S. Kolosa, A Reinforcement Learning Approach to Spacecraft Trajectory Optimization, Tech. Rep. Dissertations 3542, Western Michigan University, 2019.

APPENDIX N

Glossary of acronyms

Table N.1 Table of acronyms.

Acronym	Meaning
A/D	Analog-to-Digital
ACS	Attitude Control System
ADCS / ADACS	Attitude Determination and Control System
ADLT	Advanced Discriminating LADAR Technology
ADS	Attitude Determination System
AKM	Apogee Kick Motor
AOCE	Attitude and Orbit Control Electronics
AOCS	Attitude and Orbit Control System
APL	Applied Physics Laboratory
APS	Advanced Pixel Sensor
BOL	Beginning-of-Life
BSS	Boeing Satellite Systems
CAD	Computer-Aided Design
CAN	Controller Area Network
CCD	Charge-Coupled Device
CEP	Circular Error Probability
CFAR	Constant False-Alarm Rate
CID	Charge-Injection Device
CMG	Control-Moment Gyro
CMOS	Complementary Metal Oxide Semiconductor
CSM	Apollo Command Service Module
CSS	Coarse Sun Sensor
D/A	Digital-to-Analog
DUT1	Universal Time Correction
ECI	Earth-Centered Inertial (coordinate frame)
EF	Earth-Fixed (coordinate frame)
EHT	Electrothermal Hydrazine Thruster
EM	Engineering Model
EOL	End-of-Life
ESA	Earth-Sensor Assembly
ESA	European Space Agency

continued on next page

Table N.1 (*continued*)

Acronym	Meaning
ESTEC	European Space Research and Technology Centre
FDI	Fault Detection and Isolation
FDIR	Fault Detection Isolation and Recovery
FGS	Fine Guidance Sensor
FM	Flight Model
FOG	Fiber-Optic Gyro
FORTRAN	Formula Translation
FSS	Fine Sun Sensor
GE	General Electric Corporation
GGS	Global Geosciences Spacecraft
GN&C	Guidance Navigation and Control
GPS	Global Positioning System
GSE	Government Supplied Equipment
HRG	Hemispherical Resonating Gyro
HSA	Horizon Sensor Assembly
HSE	Horizon Sensor Electronics
HSS	Honeywell Satellite Systems
HST	Hubble Space Telescope
HT	Hall Thruster
I&T	Integration and Test
IEEE	Institute of Electrical and Electronics Engineers
IMPEHT	Improved EHT
IRAS	Infrared Astronomy Satellite
IRCS	Integrated Radar and Communications Subsystem
IRES	Infrared Earth Sensor
IRS	Infrared Sensor
ISL	Intersatellite Link
ISS	International Space Station
IUS	Interim Upper Stage
J2000	Mean of Aries 2000 Reference
JD	Julian Date
JHUAPL	Johns Hopkins University Applied Physics Laboratory
JPL	NASA Jet Propulsion Laboratory
JWST	James Webb Space Telescope
LAE	Liquid Apogee Engine
LEO	Low-Earth Orbit
LIDAR	Light Detection and Ranging
LQ	Linear Quadratic
LQE	Linear Quadratic Estimator
LQG	Linear Quadratic Gaussian
LM	Lockheed Martin

continued on next page

Table N.1 (*continued*)

Acronym	Meaning
LM	Apollo Lunar Module
LSB	Least-Significant Bit
LTR	Loop-Transfer Recovery
LVLH	Local Vertical/Local Horizontal (coordinate frame)
L2	Second Lagrange Point
MAP	Microwave Anisotropy Probe
MEMS	Micro-Electro-Mechanical System
MIL-STD	Military Standard
MIMO	Multiinput–Multioutput
MJD	Modified Julian Date
MMS	Magnetospheric Multiscale
MOL	Middle-of-Life
MRC	Measurement Reference Cube
MSB	Most-Significant Bit
MTBF	Mean Time Between Failures
MWA	Momentum-Wheel Assembly
NON	Negative Orbit Normal
NORAD	North American Air Defense Command
OMS	(Space Shuttle) Orbital Maneuvering System
OSC	Orbital Sciences Corporation
PD	Proportional-Derivative (controller)
PI	Proportional-Integral (controller)
PID	Proportional-Integral-Derivative (controller)
PON	Positive Orbit Normal
PPT	Pulsed Plasma Thruster
PRF	Pulse-Repetition Frequency
PSS	Princeton Satellite Systems
QFT	Quantitative Feedback Theory
RC	Reference Cube
RCA	Radio Corporation of America
RCS	Reaction Control System
REA	Rocket Engine Assembly
RF	Radio-Frequency
RG	Rate Gyro
RIG	Rate-Integrating Gyro
RMA	Rate-Measuring Assembly
RPY	Roll, Pitch, and Yaw
RWA	Reaction-Wheel Assembly
SCAT	Secondary Combustion Augmentation Thrusters
SEM	Sun-Earth-Moon
SHELS	Shuttle Hitchhiker Ejection Launch System

continued on next page

Table N.1 (*continued*)

Acronym	Meaning
SI	Systeme International
SISO	Single-Input–Single-Output
SPM	Spin-Precession Maneuver
SSA	Sun-Sensor Assembly
TAI	International Atomic Time
TDB	Barycentric Dynamical Time
TDRS	Tracking and Data-Relay Satellite
TDT	Terrestrial Dynamical Time
UKF	Unscented Kalman Filter
UT1	Universal Time 1
UTC	Universal Time Coordinated

Index

T

V