The background of the cover features a dark blue space-like background filled with numerous small, colorful glowing dots representing stars or particles. Several bright, multi-colored light rays (blue, green, yellow, red) radiate from the bottom left, creating a sense of motion and depth.

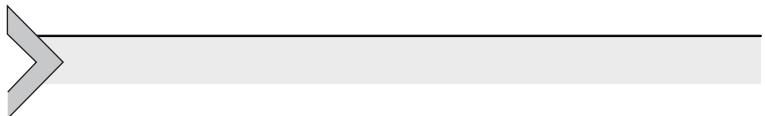
Edited by

Vincenzo Pesce, Andrea Colagrossi,
and Stefano Silvestrini

Modern Spacecraft Guidance, Navigation, and Control

From System Modeling to AI and
Innovative Applications

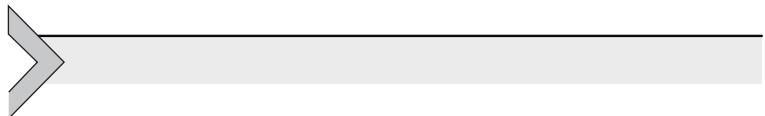




MODERN SPACECRAFT

**GUIDANCE,
NAVIGATION, AND
CONTROL**

This page intentionally left blank



MODERN SPACECRAFT GUIDANCE, NAVIGATION, AND CONTROL

FROM SYSTEM MODELING TO AI AND
INNOVATIVE APPLICATIONS

Edited by

VINCENZO PESCE

Airbus D&S, Advanced Studies Department, Toulouse, France

ANDREA COLAGROSSI

*Politecnico di Milano, Aerospace Science and Technology
Department, Milan, Italy*

STEFANO SILVESTRINI

*Politecnico di Milano, Aerospace Science and Technology
Department, Milan, Italy*



ELSEVIER

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

Copyright © 2023 Elsevier Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. Details on how to seek permission, further information about the Publisher's permissions policies and our arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: www.elsevier.com/permissions.

This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

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: 978-0-323-90916-7

For information on all Elsevier publications visit our website
at <https://www.elsevier.com/books-and-journals>

Publisher: Matthew Deans

Acquisitions Editor: Chiara Giglio

Editorial Project Manager: Sara Greco

Production Project Manager: Surya Narayanan Jayachandran

Cover Designer: Christian J. Bilbow

Typeset by TNQ Technologies



Working together
to grow libraries in
developing countries

www.elsevier.com • www.bookaid.org

Contents

<i>List of contributors</i>	<i>xi</i>
<i>Biography</i>	<i>xiii</i>

PART 0 Introduction

1. Introduction	3
Vincenzo Pesce, Andrea Colagrossi and Stefano Silvestrini	
Modern spacecraft GNC: what, why, how, for whom?	3
A brief historical review of classical spacecraft GNC	10
GNC terminology	13
GNC architecture: from requirements to preliminary design	15
Notation rules	38
References	42

PART 1 Fundamental GNC tools

2. Reference systems and planetary models	45
Andrea Colagrossi, Stefano Silvestrini and Vincenzo Pesce	
Earth and planetary models	46
Coordinate reference systems	53
Coordinate transformations	63
Time	69
What is relevant for GNC?	73
References	75
3. The space environment	77
Andrea Capannolo, Emanuele Paolini, Andrea Colagrossi, Vincenzo Pesce and Stefano Silvestrini	
Perturbation sources	78
External perturbations	79
External perturbations modeling guidelines	97
Internal perturbations	100
Internal perturbations modeling guidelines	123
What is relevant for GNC?	124
References	126

4. Orbital dynamics	131
Andrea Capannolo, Stefano Silvestrini, Andrea Colagrossi and Vincenzo Pesce	
Two-body problem	132
Three-body problem	161
Irregular solar system bodies	170
Relative orbital dynamics	174
References	204
5. Attitude dynamics	207
Aureliano Rivolta, Andrea Colagrossi, Vincenzo Pesce and Stefano Silvestrini	
Attitude kinematics	208
Attitude dynamics	227
Three-body problem attitude dynamics	248
Relative attitude dynamics	249
Multibody spacecraft dynamics	250
References	252
6. Sensors	253
Andrea Colagrossi, Vincenzo Pesce, Stefano Silvestrini, David Gonzalez-Arjona, Pablo Hermosin and Matteo Battilana	
Sensor modeling for GNC	254
Sensor faults	275
Orbit sensors	276
Attitude sensors	286
Inertial sensors	306
Electro-optical sensors	322
Altimeters	330
References	334
7. Actuators	337
Andrea Colagrossi, Lisa Whittle, Vincenzo Pesce, Stefano Silvestrini and Matteo Battilana	
Actuator modeling for GNC	338
Thrusters	344
Reaction wheels	354
Control moment gyros	366
Magnetorquers	369
References	375

PART 2 Spacecraft GNC

8. Guidance	381
Thomas Peters, Stefano Silvestrini, Andrea Colagrossi and Vincenzo Pesce	
What is guidance?	381
On-board versus ground-based guidance	382
Guidance applications	385
Guidance implementation best practices	438
References	438
9. Navigation	441
Vincenzo Pesce, Pablo Hermosin, Aureliano Rivolta, Shyam Bhaskaran, Stefano Silvestrini and Andrea Colagrossi	
What is navigation?	441
On-board versus ground-based navigation	443
Sequential filters	445
Batch estimation	461
Absolute orbit navigation	470
Absolute attitude navigation	488
Relative navigation	504
Image processing techniques	507
Navigation budgets	528
Navigation implementation best practices	533
References	540
10. Control	543
Francesco Cavenago, Aureliano Rivolta, Emanuele Paolini, Francesco Sanfedino, Andrea Colagrossi, Stefano Silvestrini and Vincenzo Pesce	
What is control?	543
Control design	545
Review of control methods	592
Control budgets	618
Control implementation best practices	626
References	629
11. FDIR development approaches in space systems	631
Massimo Tipaldi, Stefano Silvestrini, Vincenzo Pesce and Andrea Colagrossi	
FDIR in space missions, terms, and definitions	633
Current FDIR system development process and industrial practices	637

FDIR system hierarchical architecture and operational concepts	639
FDIR system implementation in European Space missions	641
FDIR system verification and validation approach	642
FDIR concept and functional architecture in GNC applications: a short overview	642
References	645
12. GNC verification and validation	647
Francesco Pace, Emanuele Paolini, Francesco Sanfedino, Daniel Alazard, Andrea Colagrossi, Vincenzo Pesce and Stefano Silvestrini	
Why it is important?	648
Statistical methods	652
MIL test	656
SIL/PIL test	667
HIL test	677
In-orbit test	682
References	683
13. On-board implementation	685
David Gonzalez-Arjona, Vincenzo Pesce, Andrea Colagrossi and Stefano Silvestrini	
Spacecraft avionics	686
On-board processing avionics	694
On-board implementation alternatives	700
On-board implementation and verification	705
References	711
PART 3 AI and modern applications	
14. Applicative GNC cases and examples	715
Stefano Silvestrini, Andrea Colagrossi, Emanuele Paolini, Aureliano Rivolta, Andrea Capannolo, Vincenzo Pesce, Shyam Bhaskaran, Francesco Sanfedino and Daniel Alazard	
AOCS design	717
Orbital control system	730
Attitude control system	742
Relative GNC	775
On-board sensor processing	791

Irregular solar system bodies fly around	803
GNC for planetary landing	806
References	814
15. Modern Spacecraft GNC	819
<i>Stefano Silvestrini, Lorenzo Pasqualetto Cassinis, Robert Hinz, David Gonzalez-Arjona, Massimo Tipaldi, Pierluigi Visconti, Filippo Corradino, Vincenzo Pesce and Andrea Colagrossi</i>	
AI in space—Introduction	821
Artificial intelligence and navigation	867
Validation of AI-based systems	883
Reinforcement learning	890
AI use cases	906
AI on-board processors	923
Innovative techniques for highly autonomous FDIR in GNC applications	925
Small satellites/CubeSats	938
References	971
Further reading	981
16. Mathematical and geometrical rules	983
<i>Andrea Capannolo, Aureliano Rivolta, Andrea Colagrossi, Vincenzo Pesce and Stefano Silvestrini</i>	
Matrix algebra	983
Vector identities	991
Quaternion algebra	994
Basics of statistics	1000
ECI-ECEF transformation	1002
References	1006
17. Dynamical systems theory	1007
<i>Francesco Cavenago, Andrea Colagrossi, Stefano Silvestrini and Vincenzo Pesce</i>	
State-space models	1007
Discrete-time systems	1009
Transfer functions	1011
References	1015

18. Autocoding best practices	1017
Francesco Pace, Vincenzo Pesce, Andrea Colagrossi and Stefano Silvestrini	
List of main architectural and implementation rules	1017
Reference	1026
Index	1027

List of contributors

Daniel Alazard

ISAE-SUPAERO, Toulouse, France

Matteo Battilana

OHB Italia S.p.A., Milan, Italy

Shyam Bhaskaran

NASA Jet Propulsion Laboratory, Pasadena, CA, United States

Andrea Capannolo

Politecnico di Milano, Milan, Italy

Lorenzo Pasqualetto Cassinis

TU Delft, Delft, the Netherlands

Francesco Cavenago

Leonardo, Milan, Italy

Andrea Colagrossi

Politecnico di Milano, Milan, Italy; Airbus D&S Advanced Studies, Toulouse, France

Filippo Corradino

Tyvak International, Turin, Italy

David Gonzalez-Arjona

GMV Aerospace & Defence, Madrid, Spain

Pablo Hermosin

Deimos Space, Madrid, Spain

Robert Hinz

Deimos Space, Madrid, Spain

Francesco Pace

GMV Aerospace & Defence, Madrid, Spain

Emanuele Paolini

D-Orbit, Fino Mornasco, Italy

Vincenzo Pesce

Airbus D&S Advanced Studies, Toulouse, France

Thomas Peters

GMV Aerospace & Defence, Madrid, Spain

Aureliano Rivolta

D-Orbit, Fino Mornasco, Italy

Francesco Sanfedino

ISAE-SUPAERO, Toulouse, France

Stefano Silvestrini

Politecnico di Milano, Milan, Italy

Massimo Tipaldi

University of Sannio, Benevento, Italy

Pierluigi Visconti

Tyvak International, Turin, Italy

Lisa Whittle

Asteroid Exploration, Leiden, the Netherlands

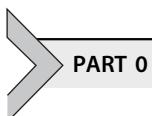
Biography

Dr. Vincenzo Pesce is a guidance, navigation, and control (GNC) engineer at Airbus D&S Advanced Studies Department in Toulouse. Prior to joining Airbus, he worked for GMV, Spain. He holds a PhD in Aerospace Engineering from Politecnico di Milano with a thesis titled “Autonomous Navigation for Close Proximity Operations around Uncooperative Space Objects.” During his studies, he spent a research period at NASA—Jet Propulsion Laboratory (2017) and at the University of Florida (2015). He is author or coauthor of about 30 scientific publications on GNC, autonomous navigation, small-body exploration, and microgravity experiments in international journals and conference proceedings. He received the prestigious Leonardo Committee Graduation Award and the Guido Horn D’Arturo Award for his research on vision-based autonomous navigation. He has been involved in several international projects in collaboration with European companies and agencies. Recently, he has been working on the development of GNC algorithms for the Mars Sample Return mission and for the ESA’s European Large Logistics Lander project. His current research interests include autonomous GNC for proximity operations, rendezvous and landing, vision-based navigation, and GNC innovative methods.

Dr. Andrea Colagrossi is an assistant professor of flight mechanics at the Aerospace Science and Technology Department of Politecnico di Milano. He holds a PhD in Aerospace Engineering with a thesis on “Absolute and Relative 6DOF Dynamics, Guidance and Control for Large Space Structures in Cislunar Environment.” He earned his MSc degree in Space Engineering in 2015 from Politecnico di Milano, with a double degree from Politecnico di Torino. In 2012, he obtained his BSc degree in Aerospace Engineering at Politecnico di Torino. He was a visiting researcher at Deimos Space (Spain) and Purdue University (Indiana, USA). He has been involved in several national and international-funded projects, collaborating with Italian and European companies, institutions, and agencies, working on development studies about GNC for modern applications, such as nanosat constellations for astrophysical observations, rendezvous in cislunar environment, and proximity operations for active debris removal. He is author and coauthor of about 30 scientific publications in international journals and conference proceedings on GNC, small space systems, and non-Keplerian dynamics. His main research interests are spacecraft GNC

and system engineering for advanced small satellite applications, with a focus on effective GNC implementation with limited hardware resources, innovative GNC techniques, and autonomous failure and contingency modes management.

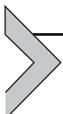
Dr. Stefano Silvestrini is a postdoctoral researcher at the Aerospace Science and Technology Department of Politecnico di Milano. He obtained his PhD cum laude with a thesis titled “AI-augmented Guidance, Navigation and Control for Proximity Operations of Distributed Systems.” He earned his MSc degree in Aerospace Engineering in 2017 from TU Delft. In 2016, he worked as a trainee for Airbus D&S in Munich. In 2015, he earned his BSc degree in Aerospace Engineering at the Università degli Studi di Padova. During his BSc, he spent a six-month research period at the College of Aerospace Engineering of Boston University under awarded scholarship. He has been involved in national and EU/ESA-funded projects for developing nanosat constellation for science observation, mission analysis, and system design for fractionated space architecture and artificial intelligence (AI) for spacecraft GNC. He has worked as a teaching assistant for several courses throughout his MSc and PhD career. His research interests include the development of AI algorithms for autonomous GNC in distributed space systems and proximity operations, particularly tailored for embedded applications in small platforms.



PART 0

Introduction

This page intentionally left blank

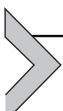


Introduction

Vincenzo Pesce¹, Andrea Colagrossi², Stefano Silvestrini²

¹Airbus D&S Advanced Studies, Toulouse, France

²Politecnico di Milano, Milan, Italy



Modern spacecraft GNC: what, why, how, for whom?

Ever since the beginning of human existence, humans had to rely on basic tools, or simply on their intuition, to “guide” themselves throughout their habitat, developing instruments to “determine their location” and to “act” accordingly. The concept of Guidance, Navigation, and Control (GNC), even if in a different fashion, was familiar to the first nomadic tribes and to the following pioneers of human exploration. Still, it is true that everyone approached the GNC problem in his own life, by simply going outside home and walking.

Spaceflight has been one of the most revolutionary shades of human exploration, allowing men and automated vehicles to escape Earth’s gravity, turning imagination and wonder into reality. The technology of satellites and space vehicles has rapidly evolved since the first signal received from the Sputnik 1 (Спутник 1) or the first awe-inspiring images from the Moon surface, becoming now capable to land man-made objects on planets, comets, and asteroids, or to deploy constellations of orbiting bodies, or to make two spacecraft encounter while traveling thousands of kilometers per second.

In performing all these activities, one of the original problems of humanity came back under a new perspective: spacecraft GNC. Even in space, one needs to know the path, the current location, and the steps to stay on the right track. In these regards, spacecraft GNC has followed a quick technological evolution, drawing inspiration from the first Earth-based methods, such as sextants, chronometers, and landmarks, to rapidly evolve acquiring the peculiarities of the space environment. Then, with the arising of powerful computers and advanced techniques, this branch of spacecraft technology made giant leaps toward autonomy and high performances. Nowadays, it is at the dawn of a new era, with artificially intelligent machines starting to help space explorers to travel across the solar system.

This book aims at presenting an updated and comprehensive set of theory and applications about spacecraft GNC, from fundamentals to advanced concepts, including modern artificial intelligence (AI)-based architectures, with focus on software and hardware practical applications. In fact, one of the main purposes of this book is to discuss updated GNC design and validation processes, from requirements to final system implementation. This book is focused on delivering a coherent path to design a spacecraft GNC system, starting from theory and digging the critical steps toward implementation. In other words, the book is intended to provide the necessary theory, omitting redundant derivations, already present in literature, from the specific perspective of spacecraft systems.

The topics discussed in the book are typically found scattered in a vast pool of literature; or they are limited to academic theoretical manuals, which have been published quite few years ago; or they are available from documentation and technical notes, retrieved from international standards, which are not easy to be understood maintaining a general overlook on the whole problem. Moreover, GNC is typically regarded as the convergence of the three individual entities: Guidance, Navigation, and Control. This is partially valid; nevertheless, a holistic approach in the design of the system is lacking in book-type literature.

The book starts from a revision of the basic tools and theoretical background required to understand how a spacecraft moves in space and what are the elements a GNC system requires to be operative. Then, the main GNC blocks are introduced, describing their role within the overall ensemble, and explaining some theoretical foundations to understand the methods and technologies of spacecraft GNC. The discussion begins from the basic knowledge needed to understand the GNC system design process, and it is integrated with the steps driving from the requirements to the system implementation and verification. Finally, spacecraft GNC applications and examples are discussed, together with a broad survey on modern techniques, methods, and scenario, including an extensive chapter on the role of AI in modern spacecraft GNC.

This book does not want to provide an in-depth theoretical description of GNC foundations, but it offers a unique perspective on design and verification processes, practical applications, and novel, cutting-edge, techniques.

A unique text to understand the fundamentals of modern spacecraft GNC design, from theory to applications, which can guide both students, young professionals, and experts, is currently missing. The practical and

applicative focus of the book makes it appealing to junior and graduate aerospace engineers. Moreover, the discussion on modern and advance techniques should make it a reference updated handbook. In fact, spacecraft system engineers and attitude and orbit control system (AOCS)/GNC specialists may want to use it to be updated on the latest promising advancements.

In summary, the book may address different categories of readers: developers, researchers, young professionals, Ph.D. students, designers in the field of spacecraft GNC; AOCS and system engineers; Assembly Integration and Verification/ Assembly Integration and Test (AIV/AIT) technicians; professionals in the field of estimation and control sciences who want to approach the space sector; experts who are looking for an updated discussion on modern problems and applications.

Book content

Following this introductory part, the book has three main parts plus an appendix section.

The first Part 1 – *Basic Tools* offers an overview of the main theoretical and fundamental concepts supporting spacecraft GNC. In particular, the used reference systems and planetary models are detailed along with the most important elements of the space environment. An overview of the main orbital and attitude dynamical models is provided. Subsequently, sensors and actuators are described, with particular focus on common GNC-related issues and on the most relevant modeling principles. With more details:

Reference Systems and Planetary Models contains the models to describe planetary geometry, with a brief introduction to position representation methods; the main coordinate reference systems; the methods to transform the coordinates from one reference system to another one; the primary time conventions and scales, including the concept of Julian dates; a final section summarizing the most relevant aspects for the GNC system.

The *Space Environment* chapter describes the main perturbation sources influencing the spacecraft dynamics, dividing between external perturbations (i.e., nonspherical gravitational, magnetic field, atmospheric drag, solar radiation pressure, and third-body) and internal ones (i.e., flexibility and sloshing, electromagnetic disturbances, internal vibrations, thermal snap, parasitic forces and torques due to thruster firing and plume impingement); the main guidelines to model the perturbation contributions; the concepts relating external and internal perturbations with the GNC design.

The chapter on *Orbital Dynamics* introduces the two-body problem and the three-body problem, with the most useful elements for a GNC engineer; the gravitational environment around irregular solar system bodies; the relative orbital dynamics.

The *Attitude Dynamics* chapter presents the fundamental rules and methods to deal with attitude kinematics; attitude dynamics, including a discussion on the inertia properties of the spacecraft and on the main equations to deal with significant attitude motions; relative attitude dynamics; multi-body spacecraft dynamics.

The chapter about *Sensors* discusses the main sensor modeling concepts for spacecraft GNC, including some elements of metrology, and the basic principles about statistics and random variables; orbit sensors; attitude sensors, inertial sensors; electro-optical sensors; altimeters.

The chapter about *Actuators* discusses the main actuator modeling principles for spacecraft GNC; thrusters; reaction wheels; control moment gyros; magnetorquers.

Part 2 – Spacecraft GNC represents the core of the book, in which the main techniques and methods for spacecraft GNC are presented and deeply analyzed. Fault detection isolation and recovery methods, GNC verification and validation tools, and on-board practical implementation are detailed with particular emphasis. Specifically:

The *Guidance* chapter contains the different implementation philosophies of on-board and ground-based guidance, the formal discriminant between AOCS and GNC systems; the guidance system design process, including two applicative cases of rendezvous guidance and attitude guidance; the most relevant guidance implementation best practices.

The *Navigation* chapter describes the main navigation filtering techniques, including sequential and batch filters; absolute orbit navigation, including the basics of GNSS, pulsars, ground-based orbit determination techniques; absolute attitude navigation; relative navigation; image processing techniques; the influence of navigation budgets on the overall GNC chain; the most relevant navigation implementation best practices.

The *Control* chapter discusses the control design process, including both the design in the state space and in the frequency domain; an introduction to nonlinear control design; Proportional-Integral-Derivative control methods; Linear Quadratic Regulator methods; robust control fundamentals, including Model Predictive Control and Sliding Mode Control; the control budgets; the most relevant control implementation best practices.

The chapter about *FDIR Development Approaches in Space Systems* presents technical solutions and industrial processes used by the space engineers to design, develop, test, and operate health management systems, also known as Failure Detection, Isolation, and Recovery (FDIR) systems.

The *GNC Verification and Validation* chapter introduces the main industrial process to verify and validate a GNC system, including Model-in-the-Loop, Software-in-the-Loop, Processor-in-the-Loop, Hardware-in-the-Loop, In-Orbit testing activities.

The *On-board Implementation* chapter presents a brief overview on the final implementation of the GNC algorithms and functions in the on-board avionics' modules, with the associated data interfaces; the main technologies for modern processing devices, such as general-purpose Processors/Microcontrollers, Digital Signal Processors, Graphical Processing Units, Field Programmable Gate Array, or specific ad-hoc electronic circuits.

The last main Part 3 – *AI and Modern Applications* introduces the most advanced solutions for spacecraft GNC. The fundamentals of AI theory and some cutting-edge spacecraft GNC applications are described in this part. A strong focus to the space environment is imposed, and the main algorithms that can benefit from AI or other modern computing techniques are considered, without an extensive, general treatment as a classical AI or computer science textbook. This part is different from a generic AI or computer science book to stress only the peculiar algorithms and aspects of modern spacecraft GNC that are applicable in the space context. In particular:

The *Applicative GNC Cases and Examples* chapter presents a set of applicative GNC examples and use cases covering the topics of GNC that are of relevance for practical and modern applications. It contains examples on AOCS design; orbital control systems; attitude control systems; relative GNC; on-board sensor processing; irregular solar system bodies fly around; planetary landing.

The *Modern Spacecraft GNC* chapter gives an overview on modern GNC techniques and methods, including an overview on AI techniques for spacecraft, on the innovative methods for GNC FDIR, and on the emerging topic of CubeSats and nanosatellites. It contains an introduction to AI in space; AI techniques for spacecraft navigation; validation of AI-based systems; reinforcement learning; AI use cases; AI on-board processors; innovative techniques for highly autonomous FDIR systems; small satellites and CubeSats GNC.

Finally, an appendix section provides a summary of the fundamentals of mathematical and geometrical rules; dynamical systems theory; Automated Code Generation (ACG) or autocoding.

How to use the book?

This book is intended to cover the entire spacecraft GNC domain, with the most updated developments, trying not to leave any concept or application unexplored. However, the topics cannot be treated with the finest details of a specialized book. Thus, the book presents the main tools and methods to deal with the covered contents and support the discussion with several literature references to further explore the topics. The reader should either have a sufficient theoretical background on the matter, or it should deepen and review those concepts that are not completely clear after a first reading of the book content.

The book is designed and structured to support the practical work and the daily life of modern spacecraft GNC/AOCS engineers, aerospace engineers, avionic developers, and AIV/AIT technicians. Thus, it can be used as a handbook, without the need of reading it from the beginning to the end. The applicative examples and the modern GNC presented in Part 3 are supported by the fundamentals of spacecraft GNC discussed in Part 2, which are developed starting from the basic tools introduced in Part 1. Hence, reading it bottom-up can be useful for an experienced GNC engineer who want to update his knowledge, while a student or a young professional is suggested to read it following the normal flow.

Most of the chapters in Part 1 and Part 2 are enriched with sections dedicated in summarizing the chapter's aspects relevant for GNC applications or to list tips and best practices to design and practically implement the GNC functions into the spacecraft system. Similarly, some modeling guidelines are clearly outlined to better support the GNC design process. The chapters about FDIR systems, verification and validation processes, and on-board implementation are self-contained, and they should be used to have a clear preliminary overview on these important concepts, often overlooked in classical textbooks about spacecraft GNC.

Part 3 contains diverse and various applicative and modern concepts, spanning the entire applicative spectrum of spacecraft GNC. Thus, it is not uncommon that the different sections of this part belong to different domains and have a broad variety of terminology and theoretical concepts. Thus, Part 3 could be read considering each section as an autonomous block, performing the necessary links to the previous parts and to the appendices of the book.

What is not contained in this book?

The book is intended to be a practical and useful support to the work and to the knowledge of spacecraft GNC/AOCS engineers, aerospace engineers, avionic developers, and AIV/AIT technicians. Thus, it is not designed to provide all the basic and theoretical concepts behind each of the covered contents. Moreover, the reader is expected to have a solid knowledge on mathematics, physics, dynamics, and basics of space systems.

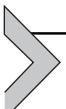
The description of orbital and attitude dynamics is constrained to the most relevant aspects useful for the GNC design. Consequently, few topics about orbit and attitude dynamics have been omitted, mainly because of length restrictions. All the landmarks and milestones of the GNC design and verification are included, but the details are sometimes just hinted in order to avoid a lengthy and cumbersome discussion that would not fit with the handbook purposes of this book. In all these cases, invitation for the reader to deepen those aspects on existing literature references is included.

The book does not contain extensive discussion on theoretical aspects of spacecraft dynamics and environment modeling. Furthermore, it overlooks the theoretical details on sensors and actuators, focusing more on the practical aspects about their integration in the GNC design (e.g., calibration, testing, and numerical modeling). In other words, in this book, environmental terms, dynamical equations, sensors and actuators are more intended as mathematical models, rather than as the physical principles they represent.

The GNC description directly applies the methods to the spacecraft design; hence, no extensive discussion about general GNC methods is included. Moreover, mathematical derivations and proofs are limited to those cases where they are directly applicable in the design and verification processes.

The example and applicative cases are limited to those with interest in present and future mission scenario. Moreover, the focus is dedicated to modern applications that are somehow less common in classic literature about spacecraft GNC. Even in this case, proper references to classic literature examples are included. The AI section only considers techniques and methodologies that are more consolidated in spacecraft GNC design. As already said, AI is specifically limited to spacecraft GNC; thus, an extensive, general treatment as a classical AI textbook is not contained in this book. Furthermore, given the close relation to research aspects, some details on the most innovative applications are given, highlighting their current development status and their applicative limitations.

The list of covered applications is obviously not exhaustive, but contains those examples that are deemed to be more relevant for the modern challenges faced, on a day-by-day basis, by the spacecraft GNC engineers. The editors apologize from now for any missing content the reader would have desired to find in this book; the interested reader is invited to contact them to directly ask for specific suggestions on how to find and deepen its knowledge about the missing topics.



A brief historical review of classical spacecraft GNC

Despite the concepts of GNC are intrinsic with the human movements across the globe, the true ancestor of spacecraft GNC is the ancient mariner, who had to explore the world by guiding, navigating, and controlling the motion of an artificial object through a vast environment, still unexplored, without references and landmarks. Indeed, to master the art of driving a ship across the sea, humans outlined fundamental concepts, invented methods and tools that are nowadays still applicable to the whole field of GNC. Three thousand years ago, Polynesians and Phoenicians mariners were capable to sail on the high seas, and ever since the humans have been developing celestial navigation, compass bearing, landmark ranging, route tracing, and so on. They also invented a variety of instruments and methods to accomplish these tasks. The first magnetic compass used in navigation applications is dated back to the 11th century, the mariner's astrolabe to the 14th century, the marine chronometer and the sextant to the 17th and 18th century. Similarly, the rhumb line or loxodrome navigation method was formulated and applied in the 16th century. Moreover, also the etymology of the word "*navigation*" belongs to the sea and to the mariners: it was first used in the 1530s, from Latin "*navigacionem*," which derives from "*navigatus*" meaning "to sail, sail over, go by sea, steer a ship," from "*navis*" (i.e., ship) and the root of "*agere*" (i.e., to drive). Note how at that time, and sometimes also in present days, the word navigation was encompassing the entire GNC definition.

The direct evolution of maritime navigation (i.e., maritime GNC) was the aircraft GNC, which shared several common elements with his progenitor. Indeed, airplanes move in vast regions following routes by means of navigation instruments and control systems, such as the pilot or the auto-pilot. In fact, as for the ships, the first aircrafts were controlled by men, who read the information of on-board instruments (e.g., altimeter, compass, attitude indicator, etc.) to follow the path they computed before the take-

off. Namely, the first aircraft GNC was human in the loop. A very similar evolution happened in space for spacecraft GNC.

As obvious, spacecraft moving in space share most of the fundamental characteristics with the ships and airplanes traveling across the seas and in the skies, even though the orbital conditions are even more harsh and difficult. The first spacecraft were indeed only passively controlled: launched into ballistic trajectories (i.e., the natural orbits) and with passive rotational stabilization. The first man-made object orbiting around the Earth was the Sputnik 1, launched by the Soviet Union on the fourth of October 1957. It was a simple polished metal sphere with a diameter of 58 cm, but it had no orbital and attitude control capabilities. The first American satellite was the Explorer 1, launched on the first of February 1958. It had not orbital control, and its rotational state was designed to spin the spacecraft around its elongated axis to achieve a passive attitude stabilization. However, soon after the release, the spacecraft entered in a flat spin dynamics, an undesirable rotation about an axis perpendicular to the preferred axis. Later it was discovered that this was due to energy dissipation from flexible structural elements, and this small accident motivated further developments of the theory of rigid body dynamics after nearly 200 years from the first Eulerian formulation. The spacecraft dynamics was not completely understood yet to correctly develop a GNC system.

The unmanned Luna-3 soviet space probe, which took pictures of the far side of the Moon in 1959, was the first guided spacecraft, having a very basic attitude control system. It was spin-stabilized for most of its flight, but its three-axis attitude control system was activated while taking photos. However, the need to control the spacecraft motion increased with the advent of manned spacecraft and more complex orbital operations.

The first human in space was the Soviet cosmonaut Yuri Gagarin, launched with a Vostok 1 on the April 12, 1961. Despite he was a trained air force pilot, carefully selected to accomplish his task, the mission was designed to be automatically controlled or controlled from ground. In fact, the medical doctors were not sure how a human might react to space environment, and therefore it was decided to lock the pilot's manual controls. This primitive spacecraft control system was only partially trusted by the spacecraft engineers themselves, and thus a code to unlock the controls was placed in an on-board envelope to be used in case of emergency. The code was "1-2-5," and Gagarin was anyway told about it before launch. Anyhow, this sun-seeking attitude control system, inherited from the Luna-3 mission, was correctly activated at 06:51 UTC to orient the Vostok

1 for retrofire. The manual emergency back-up was not needed, and the cold nitrogen gas thruster systems were automatically controlled.

Soon after Gagarin, the United States launched into orbit the astronaut Alan Shepard, on-board the Freedom 7 spacecraft, belonging to the project Mercury. The launch date was the fifth of May 1961, and, after the launcher separation, the automated attitude control system (i.e., Automatic Stabilization Control System) damped out any residual tumbling motion. Then, it steered the spacecraft around 180 degrees, so the retrorockets would face forward, ready for firing. Soon after that, Shepard began manually controlling the spacecraft's orientation. For redundancy purposes, the Mercury spacecraft's manual and automatic attitude control systems used a different set of control jets, and they had individual fuel supplies. When Shepard piloted the spacecraft, moving the three-axis control stick, he proportionally opened the valves of the manual jets. The system could be selectively enabled on each axis, with the automatic control taking the lead on the non-enabled axes. Shepard gradually assumed manual control, one axis at a time. At first, he took manual control of pitch, reorienting the spacecraft from its "orbit attitude" of 14 degrees nose-down pitch to the "retrofire attitude" of 34 degrees nose-down pitch, then returning to orbit attitude. He then took manual control of yaw along with pitch, yawing the spacecraft to the left and then to the right to bring it back in line. Finally, he assumed control of the roll axis as well, testing it and then restoring the spacecraft's roll to normal. Once Shepard had taken control of all three axes, he found that the spacecraft's manual response was about the same as that of the Mercury simulator [1]. Already then, the importance of setting up a proper design, verification, and testing simulator was evident.

The evolution curve of spacecraft design and on-orbit operations was extremely steep, and in a short time, the spacecraft were asked to accomplish complex tasks. The development of automated GNC systems was immediately started, to both improve the spacecraft performance and to aid the astronauts to pilot the capsules. Anyway, most of the GNC capabilities were demanded to ground or to the astronauts on-board. It is interesting to note that until the late 1970s, the attitude estimation and navigation techniques actually remained in a very underdeveloped state. Similarly, most of the guidance solutions were computed on ground and telemetered to the spacecraft.

The first orbital rendezvous between two spacecraft was accomplished by the United States astronaut Wally Schirra on December 15, 1965. The Gemini 6 spacecraft was maneuvered within 30 cm of the target spacecraft:

the Gemini 7. The rendezvous was not finalized with docking, but the spacecraft maintained the proximity condition for more than 20 min. The astronaut later described the manual rendezvous with a remarkable comment: “*Somebody said ... when you come to within three miles (i.e., 5 km), you've rendezvoused. If anybody thinks they've pulled a rendezvous off at three miles, have fun! This is when we started doing our work. I don't think rendezvous is over until you are stopped — completely stopped — with no relative motion between the two vehicles, at a range of approximately 120 feet (i.e., 37 m). That's rendezvous! Otherwise, it is the equivalent of a male walking down a busy main street with plenty of traffic whizzing by and he spots a cute girl walking on the other side. He's going 'Hey wait' but she's gone. That's a passing glance, not a rendezvous. Now if that same male can cut across all that traffic and nibble on that girl's ear, now that's a rendezvous!*”

The necessity of completely automated GNC subsystem to perform even the most complex operations, such as rendezvous and docking, is therefore evident. In fact, the risks and the costs of sending astronauts in space or the extreme ground effort to support spacecraft operations make automatic spacecraft GNC fundamental for present space applications.

Furthermore, modern spacecraft are required to accomplish these difficult tasks (e.g., rendezvous and docking, planetary and minor celestial bodies landing, on-orbit servicing, etc.), at a distance from the Earth that makes ground support almost impossible in real time or dramatically expensive. Hence, nowadays, spacecraft are required to be autonomous and make correct decisions, even in the case of unexpected circumstances. Indeed, we are currently in the epoch of autonomous and artificially intelligent spacecraft, exploring the space on Earth orbits and beyond, and the GNC system is crucial to guarantee this: it is the time of modern spacecraft GNC.



GNC terminology

The first terminological definitions we shall introduce are those immediately related with the word GNC. According to the dictionary, the GNC is a branch of engineering dealing with the “design of systems to control the movement of vehicles.” In particular:

- Guidance is referring to the determination of the desired path (i.e., “trajectory”) from the vehicle’s current location to an assigned target. It also establishes the desired changes in velocity, rotation, and acceleration to follow that course.

- Navigation is referring to the determination, at a given instant of time, of the vehicle's location, velocity, rotation, and angular rate (i.e., "state vector").
- Control is referring to the manipulation of forces and torques, by means of actuators (i.e., steering device, thrusters, etc.), needed to follow the desired path while maintaining vehicle stability.

For what concerns space engineering terminology, different names can be found to indicate similar subsystems, which are referred as GNC in this book. The definition of GNC can be casted into the set of functions in charge of targeted orbit and attitude computation, attitude and orbit determination, attitude and orbit control [2]. GNC is commonly used for the on-board segment, when the satellite position is controlled in closed loop, for instance, in case of rendezvous or formations flying. In the most general terms, the GNC functions are:

- Attitude estimation, which is referred as attitude navigation or determination in this book.
- Attitude guidance.
- Attitude control.
- Orbit control.
- Orbit estimation, which is referred as orbital navigation in this book.
- Acquisition and maintenance of a safe attitude state in emergency cases and return to nominal mission upon command.
- Real-time on-board orbital trajectory guidance and control.
- Real-time on-board relative position estimation and control, in case the mission requires it.

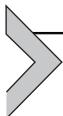
AOCS is commonly used when the orbit guidance is not performed on board, which is the case for standard low Earth orbit (LEO), medium Earth orbit, and geostationary orbit missions. However, the GNC term can be also used for the whole function, distributed between on-board and ground systems. The classical AOCSs considered here include the following functions:

- Attitude estimation, which is referred as attitude navigation or determination in this book.
- Attitude guidance.
- Attitude control.
- Orbit control.
- Orbit estimation, which is referred as orbital navigation in this book.
- Acquisition and maintenance of a safe attitude in emergency cases and return to nominal mission upon command.

Within the GNC domain, one can furtherly distinguish between AOCS and Attitude Determination Control System (ADCS). Many satellites do not

have any orbit control capabilities (e.g., without propulsion subsystem); thus, in some cases, the AOCS can be limited to the ADCS only, which features:

- Attitude estimation/determination, which is referred as attitude navigation or determination in this book.
- Attitude guidance.
- Attitude control.
- Acquisition and maintenance of a safe attitude in emergency cases and return to nominal mission upon command.



GNC architecture: from requirements to preliminary design

The GNC is a complex subsystem that extensively involves hardware and software components. Indeed, as we will see throughout the book, the GNC system interfaces sensors (cfr. [Chapter 6](#) — Sensors), on-board software (cfr. Part 2 and Part 3), and actuators (cfr. [Chapter 7](#) — Actuators). Furthermore, many implemented algorithms or, at least, their design process require a very good knowledge of the orbital and the physical environment in which the spacecraft flies.

Let us systematically decompose the GNC subsystem in the so-called GNC architecture:

- *Guidance*. The term refers to the generation of desired future plans to fulfill the mission objectives. This entails the creation of reference trajectories, both translational and rotational, that the spacecraft needs to follow. The guidance module must deliver feasible trajectories based on the spacecraft capabilities; therefore, it must take into account the spacecraft system design, as well as the constraints and the feasibility of the mission. Given its higher level of abstraction, descending directly from the mission objectives, the guidance module has been reserved to ground control for many years, nevertheless on-board guidance has recently become prominent for innovative and daring missions. The guidance module generally needs the output of the navigation to deliver inputs to the control module. Such workflow shall not be intended as dogmatic, as many different system architectures can be deployed. Certainly, the guidance module needs to know the actual state of the system to adjust future trajectories to the current system status. A comprehensive description of the guidance module is presented in [Chapter 8](#) — Guidance.

- *Navigation.* The navigation module interfaces with the sensors. It oversees the sensing of the environment to deliver the best estimate of the current state of the spacecraft, both in terms of orbital and attitude state. The navigation entails different algorithms, which are highly dependent on the sensor suite implemented on-board. The main task of the navigation is to fuse measurements from different sensors to deliver the best possible estimate, also based on the environment knowledge. In general, the navigation output represents the input to the guidance and/or control module. A comprehensive description of the navigation module is presented in [Chapter 9](#) – Navigation.
- *Control.* The control module interfaces with the actuators. It oversees the spacecraft reaction to the environment to follow the desired trajectory, based on the actual state of the spacecraft. It generally receives inputs from the guidance and navigation modules. A comprehensive description of the control module is presented in [Chapter 10](#) – Control.
- *Sensors.* The sensor suite is the ensemble of hardware in charge of sensing the environment to deliver a direct or indirect measurement of the spacecraft state, both orbital and attitude one. Sensors must be characterized and calibrated because a proper modeling of their functioning is critical for the navigation algorithms to perform well. A comprehensive description of sensors is presented in [Chapter 6](#) – Sensors.
- *Actuators.* The actuator suite is the ensemble of hardware in charge of delivering either an external torque or thrust interacting with the environment. Also, actuators must be characterized and calibrated because a proper modeling of their functioning is critical for the GNC algorithms to perform well. A comprehensive description of actuators is presented in [Chapter 7](#) – Actuators.
- *Environment.* The environment represents the ensemble of physical laws dominating the surrounding of the spacecraft. In other words, it is a mathematical representation of the natural forces and torques acting on the spacecraft. For a GNC engineer, the environment has a twofold role: on one hand, it represents a partially known entity that drives the actual system state, whose accurate modeling is critical to validate the developed algorithms; on the other hand, environmental modeling achieves different level of accuracy when designing the algorithms themselves, depending on their complexity, computational burden, etc. In other words, the former is something we cannot do much about and the spacecraft is simply reacting to it, the latter is the engineering knowledge that needs to be traded-off in each application. A comprehensive

description of the space environment, the orbital and attitude dynamics is presented in [Chapter 3 – The Space Environment](#), [Chapter 4 – Orbital Dynamics](#), and [Chapter 5 – Attitude Dynamics](#).

A schematic of a generic GNC architecture is shown in [Fig. 1.1](#).

Another important block of the GNC system is represented by a set of management routines, which fall within the so-called FDIR system. Such system oversees the malfunctioning of the GNC system with the goal of detecting the anomaly, isolating it and to compute recovery action to maintain mission operability in contingent situations. As thoroughly discussed in [Chapter 11 – FDIR Development Approaches in Space Systems](#), the FDIR involves reasoning at system level, due to the fact that spacecraft anomalies may impact the GNC system bidirectionally. Indeed, the FDIR is directly interfaced with the Mission and Spacecraft Management (MSM) system, also known as Mission and Vehicle Management (MVM), which is in charge to oversee the overall behavior of the spacecraft and its subsystems, including the GNC one, in order to achieve the mission goals, managing the system phases and modes. Adding this to our scheme of [Fig. 1.1](#), we obtain the scheme depicted in [Fig. 1.2](#).

The mission management is highly dependent on mission objectives, costs, and other collateral elements. Hence, ground control support is often

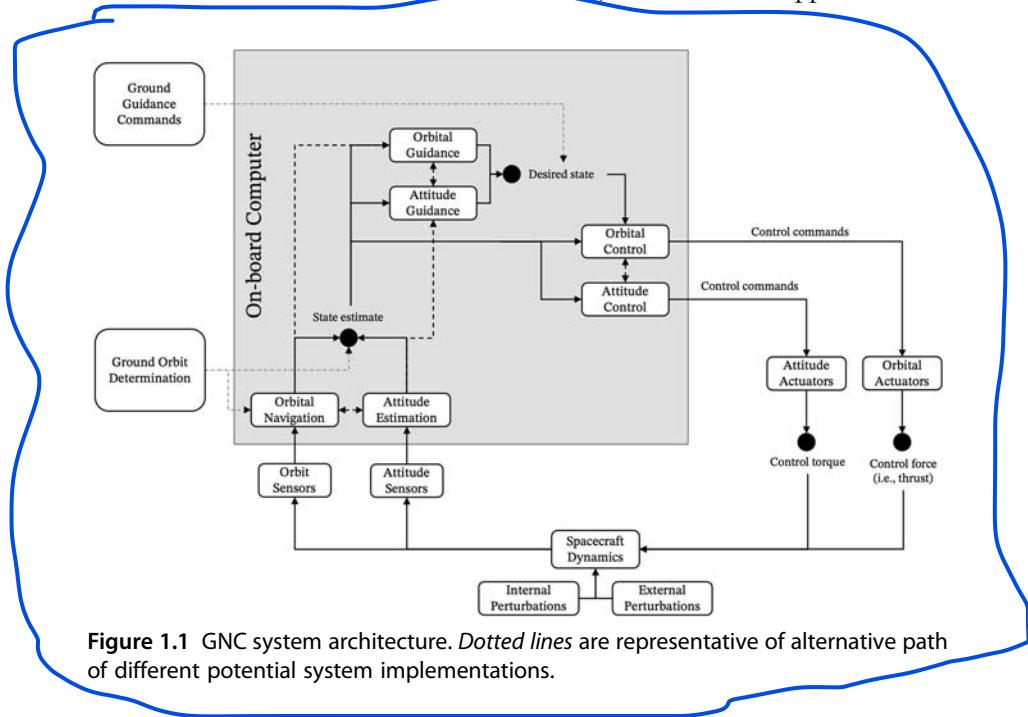


Figure 1.1 GNC system architecture. Dotted lines are representative of alternative path of different potential system implementations.

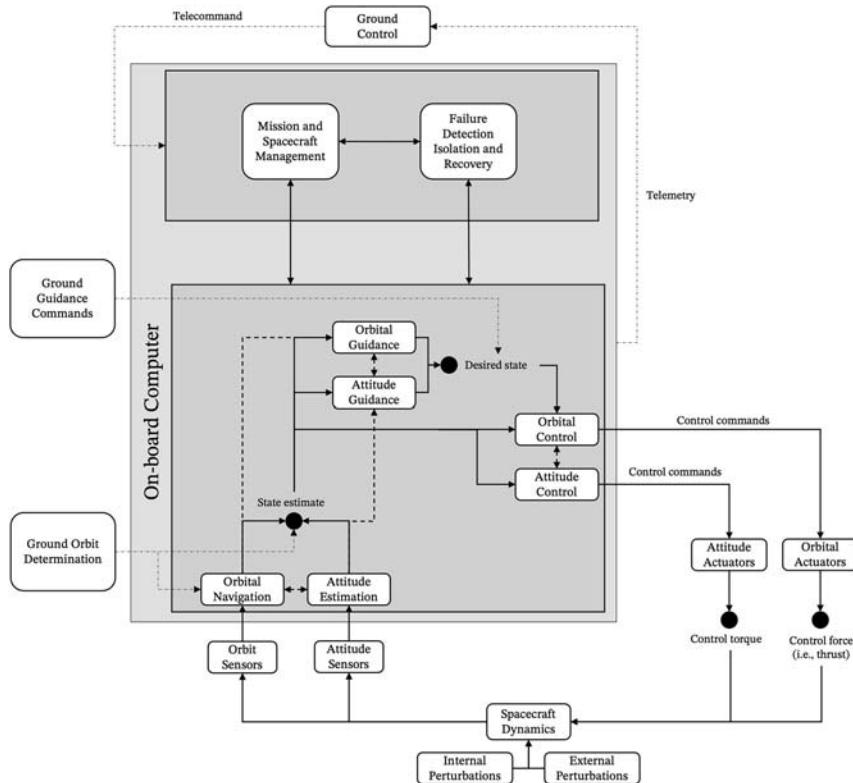


Figure 1.2 GNC-FDIR system architecture. Dotted lines are representative of alternative path of different potential system implementations.

utilized in the FDIR and MSM loop, yielding semiautonomous systems that can handle only a subset of contingent situations. As mentioned, the reader is referred to [Chapter 11 – FDIR Development Approaches in Space Systems](#) for a comprehensive description of these management blocks.

GNC subsystem design

GNC subsystem design is an iterative process. [Table 1.1](#) summarizes the typical steps in a GNC design process, with inputs and outputs that would be expected for each design step.

The first step of the GNC design process is the definition of guiding requirements based on mission goals, as shown [Table 1.1](#). The GNC requirements for the development of space programs are typically part of the *Project Requirements Document*, which is comprehensive of the entire mission. The levels of completeness and detail vary very much from project to project.

Table 1.1 GNC subsystem design steps.

Design step	Description	Inputs	Outputs
1	Subsystem requirements definition and derivation—subsystem criticalities	<ul style="list-style-type: none"> • Mission requirements • System requirements • Subsystems constraints • Mission Concept of Operations (ConOps) • Mission timeline 	GNC subsystem requirements
2	Definition of GNC modes	<ul style="list-style-type: none"> • GNC subsystem requirements • Subsystems constraints • Mission ConOps • Mission timeline 	List of different GNC modes
3	Environment and disturbance assessment	<ul style="list-style-type: none"> • Spacecraft geometry • Operational orbit • Epoch and mission timeline • Mission ConOps 	<ul style="list-style-type: none"> • Station-keeping needs • Disturbance forces (i.e., internal and external) • Disturbance torques (i.e., internal and external)
4	Sizing and selection of GNC subsystem hardware	<ul style="list-style-type: none"> • Spacecraft geometry • Operational orbit • Epoch and mission timeline • Mission ConOps • Disturbances • Subsystem constraints 	<ul style="list-style-type: none"> • Preliminary definition of the sensors suite • Preliminary definition of actuators suite • Preliminary definition of computational architecture (i.e., on-board software — OBSW — and on-board computer — OBC)

(Continued)

Table 1.1 GNC subsystem design steps.—cont'd

Design step	Description	Inputs	Outputs
5	Definition of GNC algorithm	<ul style="list-style-type: none"> • Sensors suite performance and characterization • Actuators suite performance and characterization • Subsystem performance requirements 	<ul style="list-style-type: none"> • Algorithms for GNC subsystem • Mode manager
6	Iterate from 1	Output from 1, 2, 3, 4, 5	Refined mission and GNC subsystem requirements.

This is very important for a designer that shall identify the right level of complexity and performance for the project. Let us state it as a *golden design rule*:

A designer knows that he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away.

For each mission, it is necessary to adapt the specified requirements through a complete tailoring process [2], which entails:

- To decide if a requirement is necessary, taking into account the specific functionalities required for the mission (cfr. golden design rule). For instance, if a mission requires an on-board navigation function, then the requirements dedicated to this function or to an on-board GNSS receiver are applicable.
- To adapt the numerical values of a requirement, considering the exact performances required for the mission. The designer must reason on the actual needs of the mission, without spending resources for unnecessary performance level.
- To quantify the new hardware and software development necessary for the program, which is a key factor in adapting the verification requirements.

The types of GNC requirements can be divided into [2]:

- Functional requirements.
- Operational requirements.

- Performance requirements: these requirements are often referred to characterize the GNC system in terms of the following features:
 - *Performance errors and budgets.* For a comprehensive description of these errors, please refer to [Chapter 10](#) – Control – Control Budgets.
 - *Stability.* It expresses the ability of a system submitted to bounded external disturbances to remain indefinitely in a bounded domain around an equilibrium position or around an equilibrium trajectory.
 - *Stability margins.* They are used to quantify maximum parameters excursions preserving stability properties. The most frequent stability margins, defined in classical control design (cfr. [Chapter 10](#) – Control), are the gain margin, the phase margin, the modulus margin, and, less frequently, the delay margin.
- *Agility.* It assesses the ability of a system to perform a maneuver in a given time interval, including the tranquilization phase.
- *Transient response.* It characterizes the ability of a system to reach the steady state with given maximum parameters. Common response metrics are the settling time or system overshoot.
- *Robustness.* It describes the ability of a controlled system to maintain some performance or stability characteristics in the presence of plant, sensors, actuators, and/or environmental uncertainties.
- Verification requirements.

Typically, as for other subsystems, the GNC requirements originate from mission requirements, expressed in various ways and directly linked to the final objectives of the mission. It is interesting to report that international standard claims that, in some cases, it can be preferable to keep the performance requirements expressed at mission level and not at GNC level, in order to allow the best optimization of the system [2]. An applicative use case showing the GNC design process from requirements to preliminary design is presented in [Chapter 14](#) – Applicative GNC Cases and Examples.

The GNC subsystem is a part of the whole spacecraft; thus, many requirements may also come from other spacecraft subsystems. The interfaces between subsystems are obviously critical to be assessed when dealing with system and subsystem design, especially GNC. An example of input/output relationship is given in [Fig. 1.3](#).

GNC modes

Mission objectives and requirements often imply more than one mode of operating a spacecraft. Indeed, a contextual step to the requirement generation is the identification of the GNC modes. Often, the guiding

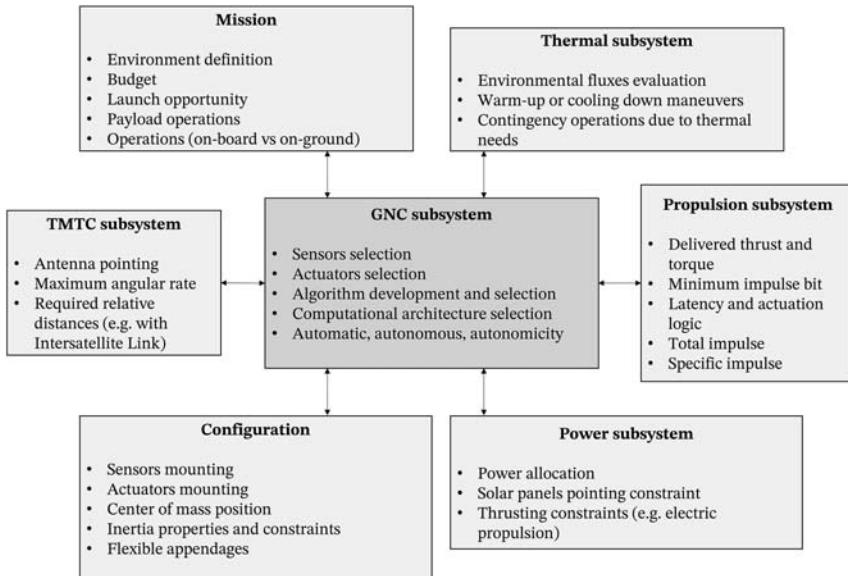


Figure 1.3 Typical system and subsystem interfaces with GNC.

requirements generally begin with a description of the control modes the GNC is expected to execute to meet those goals. This is because most of the requirements, functional, operational, and performance are dependent on the specific GNC mode involved.

- GNC modes shall be defined by evaluating mission operations against:
- *Flexibility*. It is the capacity to adapt and solve more circumstances, without ambiguities, with simple and effective configuration options.
 - *Autonomy*. It is the capacity to be operative without the need of ground intervention.
 - *Redundancy*. It is the capacity to be operative in case of the loss of some hardware components.
 - *Performance*. It is the capacity to guarantee GNC functionalities with various control authority and computational performances.

In principle, each GNC mode may rely on different sensors, actuators, navigation, guidance, and control functions. In particular, the GNC functions should be as independent and unique as possible.

Typical GNC modes are reported in [Table 1.2](#), and they are thoroughly discussed in [Chapter 14](#) – Applicative GNC Cases and Examples.

Table 1.2 Typical GNC modes.

GNC mode	Tasks	Characteristics
Commissioning	<ul style="list-style-type: none"> • Commissioning of the GNC system • Sequential start-up of components 	<ul style="list-style-type: none"> • To be used before relevant mission phases • It should include a monitoring of the system in order to be able to detect (autonomously or from ground) anomalies prior to the operative phases • No particular performance requirement is imposed • Refer to Chapter 12 — GNC Verification and Validation
Nominal	<ul style="list-style-type: none"> • It fulfills the main mission objectives • It may be comprehensive of submodes, such as image acquisition, communication pointing, proximity control 	<ul style="list-style-type: none"> • Best performing hardware is involved • Highest performance requirements
Safe	<ul style="list-style-type: none"> • Detumble the satellite • Power-saving • Stable pointing, usually toward the Sun 	<ul style="list-style-type: none"> • Reliable hardware is employed • No performance requirement is imposed (i.e., also rough navigation or control are accepted) • Must be autonomous
Standby	<ul style="list-style-type: none"> • Power positive pointing • Ground station pointing • Particular pointing required by a set of subsystems (e.g., power and thermal) • Drag drift minimization (e.g., hold point) 	<ul style="list-style-type: none"> • In general, the spacecraft is more operative with respect to safe mode • Different GNC functions with respect to Safe mode
Orbit control	<ul style="list-style-type: none"> • It performs orbital maneuvers 	<ul style="list-style-type: none"> • It can be split into inertial and relative maneuvers • Propulsion subsystem is involved • Parasitic torques coming from thrust misalignment have to be controlled
Transfer	<ul style="list-style-type: none"> • It controls the attitude and orbital state during long transfers (e.g., to get to GTO or interplanetary transfer) • Interplanetary navigation 	<ul style="list-style-type: none"> • It shares a lot with orbit control mode • It may have additional system requirements during ballistic arcs
Special	<ul style="list-style-type: none"> • It fulfills particular extended objectives of the mission 	<ul style="list-style-type: none"> • Various

System redundancy

With respect to the mode's characteristics, it is important to highlight the concept of redundancy. The harsh space environment yield degradation or failures of space missions. A typical requirement for many missions is the ability to survive and function even in the case components failures have occurred. Redundancy aims at implementing such requirement to preserve the full capabilities of the spacecraft. Redundancy can be split into *cold*, *hot*, and *active* [3]:

- *Cold redundancy design*. The secondary hardware is not operative and normally switched off until a failure on the primary component occurs. The advantage of such kind of redundancy is that the component does not require any power. Nevertheless, the latency of the system to react to the failure increases, as the secondary component needs to be turned on and, potentially, commissioned.
- *Hot redundancy configuration*. All entities are powered on with only one operating. This allows a quick recovery of the functionality whenever a failure in the primary component occurs.
- *Active redundancy configuration*. All the components are operative, and the system can continue to function without downtime or defects despite the loss of one or more entities.

Finally, if redundancy is implemented at functional level (i.e., not on specific hardware), it is often referred as *analytical redundancy*. In an analytical redundancy configuration, the function of a failed component is provided by using an entirely different component or ensemble of components, with a different set of GNC functions. Such strategy largely involves algorithm design and often requires higher computational cost.

The redundancy of multiple GNC elements (e.g., architecture including sensors, on-board computers, actuators, etc.) can be implemented in mainly two different alternative configurations, namely in a *block redundant* or in a *cross-strapped* configuration [4]. Their difference is depicted in Fig. 1.4.

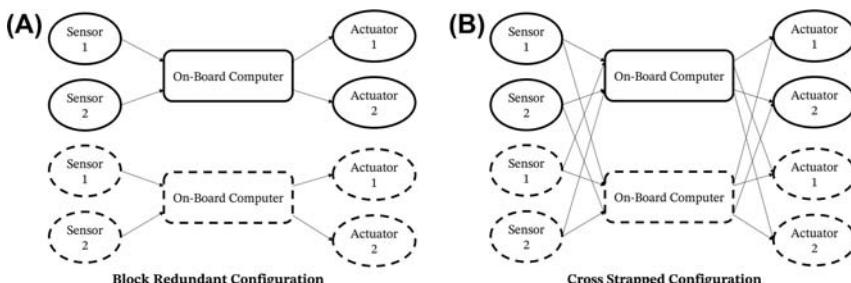


Figure 1.4 Multiple elements redundancy configuration. *Dotted lines* indicate redundant components in the nominal design.

Block redundant configuration replicates two (or more) independent fully functional pipelines. Whenever one of the components (e.g., sensor 1, OBC, or actuator 2) fails, the entire pipeline is dismissed and the redundant is used, being it in cold, hot, or active redundancy. In a cross-strapped configuration, each element is connected to both the nominal and the redundant successive element. Thus, the failure of one element in one pipeline does not prevent the elements belonging to it to maintain their operativity.

Both configurations are single-fault tolerant. Block redundant configuration has a simpler implementation, but it can withstand a second fault only if the failure occurs in the pipeline where the first one occurred. It can be seen that the number of connections linearly scale with the number of redundant paths. Cross-strapped configuration is certainly more robust at the cost of higher complexity, given the large number of additional connections, potentially unused in nominal conditions. In this case, it is important to remark that the number of functional connections exponentially scale with the number of redundant paths.

Each mission requires an accurate reliability analysis to define the proper redundancy strategy and configuration. Moreover, throughout the generation of the requirements, the definition of the modes, the selection of system redundancies and the preliminary design, the concept of trade-off is crucial. Several iterations, as detailed in [Chapter 14 — Applicative GNC Cases and Examples](#), are required to converge to the final GNC architecture.

Mission phases

The GNC system design, implementation, and operation follow the development of the standard mission phases. [Table 1.3](#) reports an overview of the different phases of the life cycle of a typical space mission.

Table 1.3 Mission life cycle.

Phase 0	Mission analysis and identification
Phase A	Feasibility
Phase B	Preliminary definition
Phase C	Detailed definition
Phase D	Qualification and production
Phase E	Utilization
Phase F	Disposal

During phase 0 and A of the mission design, prototypes of key GNC algorithms may be developed to demonstrate the feasibility of the intended approach. The GNC algorithms may be run in open loop as stand-alone functions, or they may be integrated in highly simplified simulators of the dynamics to ensure that the algorithms can sequentially run in discrete time.

During the late phase A and in the early phase B (i.e., B1) of the mission design, a full simulator containing all the dynamical effects that are relevant for the GNC design may be implemented. However, this simulator still contains simplified models of the sensors and actuators. During phase A/B1, the architecture and the most important modes of the GNC functions are identified. A preliminary implementation of the full GNC subsystem may be carried out and tested to demonstrate the feasibility of the GNC solution. Some efforts may be spent to ensure that all the functions meet the requirements on the computational load; that is to say, to demonstrate that the GNC functions can be executed within the available time. The technical requirements for the on-board software are defined at the end of phase A.

During the phases B and C, the GNC architecture and the functionalities of the GNC software are consolidated, and the engineering efforts shift toward verification and validation of the software. Code analysis tools may be used to ensure that no prohibited operations (e.g., division by zero) can occur, and again to ensure that the GNC functions meet requirements on the computational load. During these phases, the software matures from prototypes to the actual flight software. More attention is paid to the correct definition of interfaces and more detailed models of the real world, in particular sensors and actuators, are included into the GNC simulator, as more information from the hardware design of the spacecraft becomes available.

During phase D, the final version of the on-board software is integrated in the spacecraft computer for validation, verification, and testing purposes. The whole system is manufactured and integrated to be finally qualified for the launch in space. Even the on-board software undergoes qualification testing activities before launch. Finally, during phase E, the spacecraft and the GNC subsystem are utilized and operated by the operation engineers. The original software and GNC development team may be occasionally called upon to solve problems, and even patch software problems if required.

Consider the anomalies

An operational spacecraft shall be able to handle nonnominal conditions that may arise, endangering the correct operation of a given subsystem or the

overall mission. In fact, on-orbit maintenance is not yet a feasible and economically sustainable alternative for most of the space missions. For this reason, it is extremely important to design an FDIR subsystem to detect and successfully recover, in an autonomous or semiautonomous way, from faults. FDIR system development should be carried out in parallel with the mission design and it's addressed since the very beginning of the GNC development. The definition of reliability, availability, and safety objectives is strongly dependent on the designed FDIR system.

FDIR system conception and implementation for a spacecraft is an extremely complex task. This is particularly true because the analysis of spacecraft failures is not always straightforward. The cause of malfunctions could be one or more failed components, incorrect control actions, or external disturbances, which impact system operations. Usually, in order to identify the failure scenarios, reliability design analysis (i.e., risk analysis) techniques are used. In particular, Fault Tree Analysis (FTA) and Failure Mode Effects and Criticality Analysis (FMECA) are usually employed. FTA is a top-down deductive method where top-level failures are addressed and all the basic faults (i.e., failure causes), which can lead to such states, are identified. FTA helps to identify how the system or subsystem can fail and the best way to minimize the risk connected to a given failure. On the other hand, FMECA is an inductive bottom-up analysis, and it is used to compute the causal effects on the system of a given fault combination. It is primarily used to identify potential single-point failures.

FMECA is usually applied for spacecraft risk assessment and the main process steps are [5,6]:

- *Establish FMECA analysis approach.* The purpose of this phase is to define the objectives of the analysis and the mission phases to be examined. Failure modes and criticality levels are defined. Procedures to be adopted and format of the final FMECA worksheet are selected.
- *System definition.* Functional behavior, operational modes, and mission phases are identified.
- *Construct functional block diagram.* A functional block diagram of the system to be assessed is detailed. Blocks and outputs at the level of detail required for the analysis are identified.
- *Identify failure modes and effects.* All functional, hardware, software, and product single point failures are identified and eventually included into the critical items list. Critical items are those items which are reliability, mission, or safety critical.

- *Failure mode effect.* The failure effect for each mission phase and for the considered components is assessed.
- *Failure detection method.* Failure detection techniques are defined for each mission phase. Telemetry and on-board fault management can be used to this purpose.
- *Provide compensating provisions.* Failure mode compensation methods are identified for each failure mode. This is usually carried out by the on-board autonomous fault management system (i.e., FDIR) or by the ground anomaly detection and resolution control system, which implements recovery plans and procedures. These methods can include fault tolerant systems, redundancy, workaround, and alternate modes of operations.
- *Perform criticality analysis.* Each failure mode is evaluated in terms of the worst potential consequences and a severity category is assigned.
- *Documentation and reporting.* Document the analysis and summarize the results and the problems that cannot be solved by the corrective actions.
- *Recording.* Record all critical items into a dedicated table as an input to the overall project critical item list.

Once the FMECA documentation is produced, an assessment of the risk associated with each failure mode can be performed evaluating the following quantities:

- Severity number (SN): the severity of failure effects on the system.
 - 1 – catastrophic.
 - 2 – critical.
 - 3 – major.
 - 4 – minor or negligible.
- Probability number (PN): probability of occurrence of a failure mode.
 - 1 – extremely unlikely.
 - 2 – unlikely.
 - 3 – likely.
 - 4 – very likely.
- Detection number (DN): the probability of detection of a failure mode.
 - 1 – very likely.
 - 2 – likely.
 - 3 – unlikely.
 - 4 – extremely unlikely.
- Criticality number (CN): obtained as product of SN, PN, and DN (i.e., $CN = SN \cdot PN \cdot DN$) It quantifies the risk associated with a given failure mode.

Mode management

During the GNC design, the modes shall be defined together with their transitions. Indeed, once the main modes at system and GNC subsystem level are identified, a mission manager has to be designed. The role of a mission management software is to provide commands to the GNC subsystem in order to adapt its modes and configurations to a specific mission phase. In general, an on-board mission manager shall perform four main tasks [7]:

- *Monitoring.* Processing spacecraft state and status, checking for possible problems with the current plan. Given certain mission objective, the mission manager should be able to evaluate if the current state and status, propagated in time to the end of the mission phase, satisfy the given requirements.
- *Diagnosis.* Identifying problems with the current plan and configuration and taking the action of replanning or reconfiguration. This task is very similar to the FDIR objective even though mission manager diagnosis is limited to plan and configuration problems.
- *Planning.* Creating a series of commands that achieve mission objectives. This is done at the beginning of the mission or after replanning.
- *Execution.* The plan is enabled by sending sequential commands to the Guidance and Control functions. This sequencing may be time- or event-triggered.

A high-level schematic of a mode management system is reported in Fig. 1.5. Sometimes, to simplify the overall architecture, a plan is defined beforehand, and the mission manager tasks are reduced to monitoring and execution. This is a common practice for many spacecraft missions to reduce the overall mission complexity and shorten validation and verification activities.

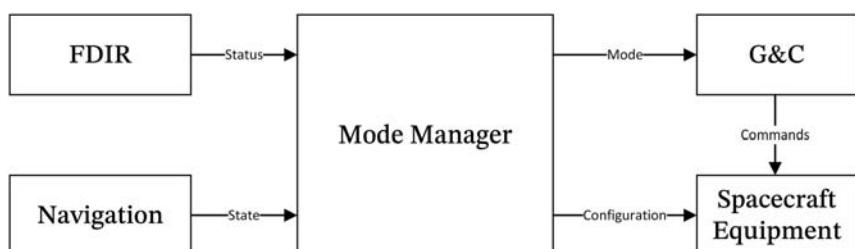


Figure 1.5 Mode manager architecture.

Mode transition and finite state machine

During the design of a mode management system, particular attention must be paid to the definition of mode transitions. In general, a good practice while implementing GNC mode transitions is to accurately design them in order to avoid:

- Logical errors (e.g., an event triggering no state).
- Ambiguous states (e.g., an event triggering multiple states).
- Transition errors (e.g., an event triggering the wrong state).
- Unreachable states (e.g., states that cannot be activated by any event).
- Loop states (e.g., states that cannot be excited after their activation).

A common solution to design GNC mode architecture and transitions is to use Finite State Machine (FSM) and visual programming. The combination of these two tools allows for a formal, simple, and visual representation of the modes of architecture and eases the operator monitoring phase. An FSM is a mathematical model describing an event-triggered system, allowing to model transitions within a fixed number of states. FSMs have the following properties:

- Can be in exactly one of a finite number of states at any given time.
- Change from one state to another (i.e., perform a transition) in response to some inputs.
- Are dynamics, discrete, finite.
- Can be represented as directed graph.

FSM can be described in Specification and Description Language, which can make these models executable for verification, validation, and testing purposes. Two major types of FSM are acceptors and transducers [8]. The main difference between these two categories of FSM is that an acceptor generates a binary output that indicates if the given input is accepted and rejected, while the transducers FSMs performs actions to generate the expected outputs. Finally, an interpreter is needed to implement and execute the FSM.

The simplicity of FSM models allows for a visual representation of states and modes transitions based on node-link diagrams (i.e., visual programming) using dataflow languages [9]. A common software used for mode transitions and management is MATLAB/Simulink Stateflow tool [10]. With this approach, each FSM can be treated as a single module and a diagram representing mode transitions through FSM can be graphically depicted, providing a very intuitive visual representation.

Automation, autonomy, and autonomicity

During the design of mode management and, in general, of spacecraft operations, different levels of autonomy can be identified. Even if we speak about autonomy while referring to a process executed without any human intervention, specific terms are defined to indicate different facets of this concept [11]:

- *Automation.* A sequence of commands is executed by software/hardware, replacing the corresponding manual process. No independent decision-making task is expected. In the case of automatic mode transitions, these steps are executed according to a given schedule (i.e., time-triggered), or after a task is completed, or after an event occurrence (i.e., event-triggered).
- *Autonomy.* Software and hardware components try to emulate the human process rather than simply replacing it. In this case, human-independent decisions are expected. In the case of mode transitions, an internal logic is designed to command the mode switch based on the received states and status of FDIR and GNC subsystem.
- *Autonomicity* (or *intelligent autonomy*). This kind of systems is expected to be able to perform self-management rather than simply self-governance, as in the case of autonomy. In other words, autonomicity can be seen as a form of autonomy specifically designed to manage the system. As an example, the goal of a system may be to detect a given phenomenon using an on-board payload [11]. The autonomy capability is to choose between different parameters to achieve this goal. However, the objective of ensuring that the system is fault-tolerant and continues to operate under fault conditions falls into the autonomicity domain rather than autonomy. Without autonomic capabilities, the spacecraft's performance degrades, and the spacecraft may be unable to recover from faults. As discussed before, in the last years, the concept of autonomicity is gaining attention for spacecraft operation and design. Autonomic systems definition derives from the human Autonomic Nervous System that is the core of most nonconscious brain activities (e.g., breathing, reflex reactions, etc.). The idea is to have a self-managing system with the following objectives [11–16]:
 - *Self-configuring.* System's ability to readjust itself automatically.
 - *Self-healing.* System's ability to recover after a fault occurs.
 - *Self-optimizing.* System's ability to measure its current performance and to improve it.
 - *Self-protecting.* System's ability to defend itself from external threats.

On-board versus ground-based

In general, a space mission consists of the ground segment and the space segment and controlling and operating a spacecraft is a task shared between the ground segment and the space segment, depending on the level of autonomy of the platform.

Fig. 1.6 shows a ground-based control system. In this case, spacecraft data obtained from the sensors are sent to ground for processing, and commands are calculated under human supervision. These commands are uploaded to the spacecraft where the commands are stored and managed by the on-board computer. The on-board computer sends the commands to the actuators at the appropriate times.

More computational power is available on-ground than on board a spacecraft. This means that the ground segment can use more complete models of the spacecraft dynamics and sensors, more sophisticated filtering techniques to obtain a guidance solution, and more detailed dynamics models for computing maneuvers. Maneuvers can be computed on ground using optimization techniques that may be prohibitively expensive to implement on-board or using techniques that require human supervision to ensure that the maneuvering solution is acceptable. In general, downloading the telemetry data requires regular ground station visibility, and the visibility

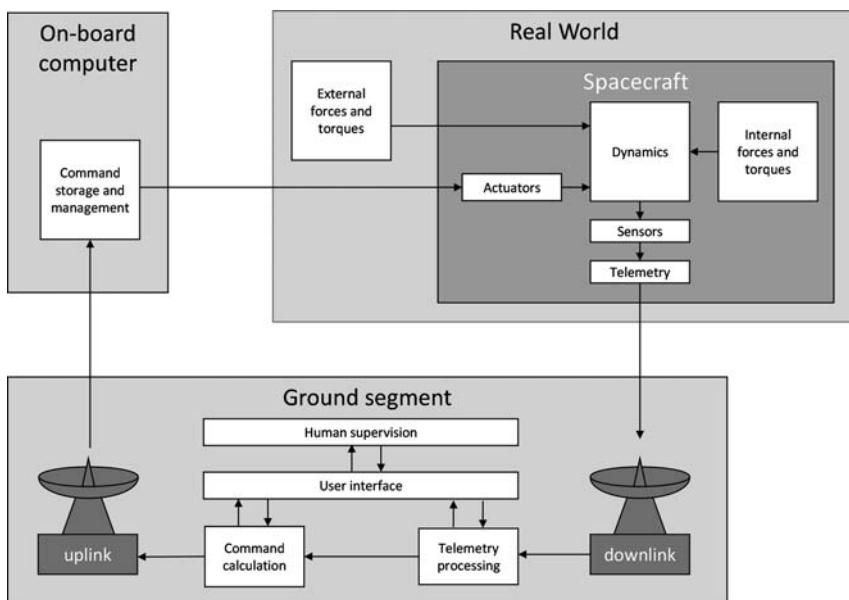


Figure 1.6 Notional ground-based system.

windows determine the frequency with which the spacecraft orbit and attitude can be determined, and commands can be uploaded. The number of ground stations that are available and their overlap determine whether and how long continuous contact is possible, and how long a spacecraft needs to be able to survive between contacts. Needless to say, the more ground stations needed for the mission, the more expensive the operations become. Likewise, if the space segment needs to be monitored continuously by human operators, then the cost of operation is high. It is possible to design the overall mission in such a way that critical operations can count on continuous supervision, while during the less critical operations, the supervision by human operators is reduced.

Another important aspect is that downloading and processing the data and generating and uploading the commands can require a significant amount of time. This is especially true for missions that take place in orbits that are further away than LEO. For example, a Mars Sample Return mission requires a higher degree of autonomy because the two-way light time can be as high as 48 min. In general, rendezvous missions may require autonomous abort capabilities for the close proximity operations as the reaction time available becomes too short for human operators to respond in time adequately [12].

For example, if robotic proximity operations are conducted with a ground operator in the loop, then it may be necessary to download video data and upload thruster and robotic arm commands. This requires a high-bandwidth link, and if the operations are critical, it must be ensured that the operations can be completed within the ground visibility window.

[Fig. 1.7](#) shows an alternative control system in which more autonomy is placed in the space segment. A navigation function processes the data coming from the sensors, a guidance function provides the reference trajectory and the feed-forward forces and torques, and the control function provides feed-back forces and torques based on inputs from the navigation and the

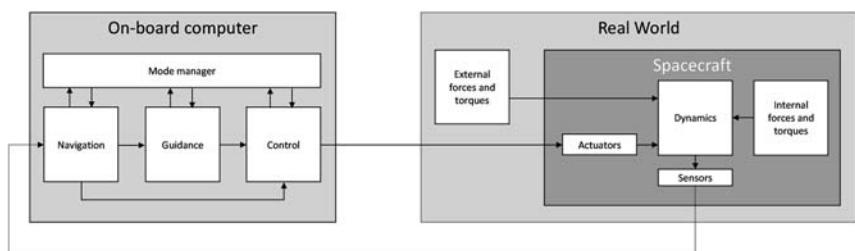


Figure 1.7 Notional on-board GNC system.

guidance function. A mode manager is in charge of managing the overall behavior of the on-board GNC system.

In recent years, microprocessors for space applications have become substantially more powerful. However, on-board computational resources are still usually limited when compared to the resources available on-ground. This means that the algorithms for on-board applications tend to be less sophisticated than the algorithms for on-ground applications.

The advantage of having greater autonomy on-board is that it allows for less frequent commanding of the spacecraft. For example, this could enable missions that require ground personnel to be present at the mission operations center only during working hours, which in turn would reduce the cost of operations. On the other hand, the increased level of autonomy also requires more time and effort to be spent on the development, verification, and validation of the software, which increases the cost of the on-board software. In general, the reduction of the cost of operation is expected to be greater than the increased cost of the software development [13].

A space system can also use a mixed approach. Some functions could be performed on-board while others are performed on-ground. For example, the time scales involved are different for the attitude dynamics, which tend to be faster, and the orbital dynamics, which tend to be slower. Functions such as target pointing for rendezvous require a fast reaction time, such that it may make sense to implement this functionality on-board, while orbital maneuvers can potentially be planned up to perhaps an hour in advance. Of course, implementing target pointing capability on-board also implies the need to implement a relative navigation function and an attitude controller on board. In addition, the level of autonomy could be dependent on the mission phase. For example, a robotic capture could be performed under full ground control, while the rendezvous operations that precede the capture are performed autonomously. The level of control the ground has can also vary. For example, the ground interactions could be limited to sending GO/NO-GO commands, or be as extensive as having full control over the spacecraft with a direct video link. Table 1.4 provides a formal description of autonomy of space systems that is based on the ECSS Space Segment Operability Standard [14]. The approach that is described in Fig. 1.7, and the sections that follow tacitly assume a level of autonomy that is broadly compatible with level E3 in Table 1.4.

Table 1.4 Autonomy levels as defined in the European ECSS space segment operability standard.

Level	Description	Functions
E1	Mission execution from ground control; limited onboard capability for safety issues	Real-time control from ground for nominal operations. Execution of time-tagged commands for safety issues
E2	Execution of preplanned, ground-defined, mission operations on-board	Capability to store time-based commands in an on-board scheduler
E3	Execution of adaptive mission operations on-board	Event-based autonomous operations. Execution of on-board operations control procedures
E4	Execution of goal-oriented mission operations on-board	Goal-oriented mission (re-)planning

Summarizing, the reasons for increasing the level of on-board autonomy are the following:

- Reduces the complexity of ground operations.
- Implies that the ground segment no longer has direct control over on-board operations that may be critical to the survival of the spacecraft [15].
- Reduces personnel requirements on ground, possibly single shift, and it may reduce propellant cost and it allows for more precise timing of on-board events [16].
- Reduces operational complexity when there are limited communication windows and/or large communication delays, such as, for example, planetary sample return missions.
- Reduces operational complexity when fast actions are required, such as, for example, detection of an imminent collision.

The European ECSS Space Segment Operability Standard provides a definition of on-board autonomy [16]: “On-board autonomy management addresses all aspects of on-board autonomous functions that provide the space segment with the capability to continue mission operations and to survive critical situations without relying on ground segment intervention.”

Table 1.5 [9] provides an overview of factors that influence the level of autonomy.

Table 1.5 Factors that influence the level of autonomy.

Assumption	Comments
Environment with high uncertainty	Three sources of uncertainty: partial observability which leads to a partial understanding of the state of the real world; nondeterminism appears in real-world domains because actions can lead to different possible states; a dynamic domain could spontaneously change its state due to external events.
Limited on-board resources	Available on-board resources, especially in terms of computing power, memory, and energy, are limited.
Limited communications	Communications to the spacecraft might be limited due to obstacles (e.g., operations inside a crater), long delays (e.g., interplanetary missions), or communication windows.
Highly complex operations	Increasing payload and platform capabilities enables the achievement of more complex missions.
Criticality	Spacecraft represent critical systems for which high safety standards must be enforced.

Verify the preliminary design

As previously explained, after the preliminary GNC design and implementation, preliminary verification and validation of the software is performed.

The software design and development process itself is shown in Fig. 1.8. In particular, for the given software development projects, the following phases can be identified:

- Definition of requirements.
- Definition of software architecture and preliminary design.
- Detailed design.
- Implementation.
- Verification.

Customer or mission requirements are encoded in the *User Requirements Document*. These requirements are flowed down to algorithm and software requirements specifications. The algorithm and software requirements are contained in the *Software Requirements Document*. Based on this document, a preliminary design of the software, including the software architecture, is performed, followed by a detailed design of the software and the algorithms. During each of these steps, a *Software Verification and Validation plan* is created and maintained, such that all the algorithms can be tested and verified. After the algorithms are coded, the software is tested. The first tests are

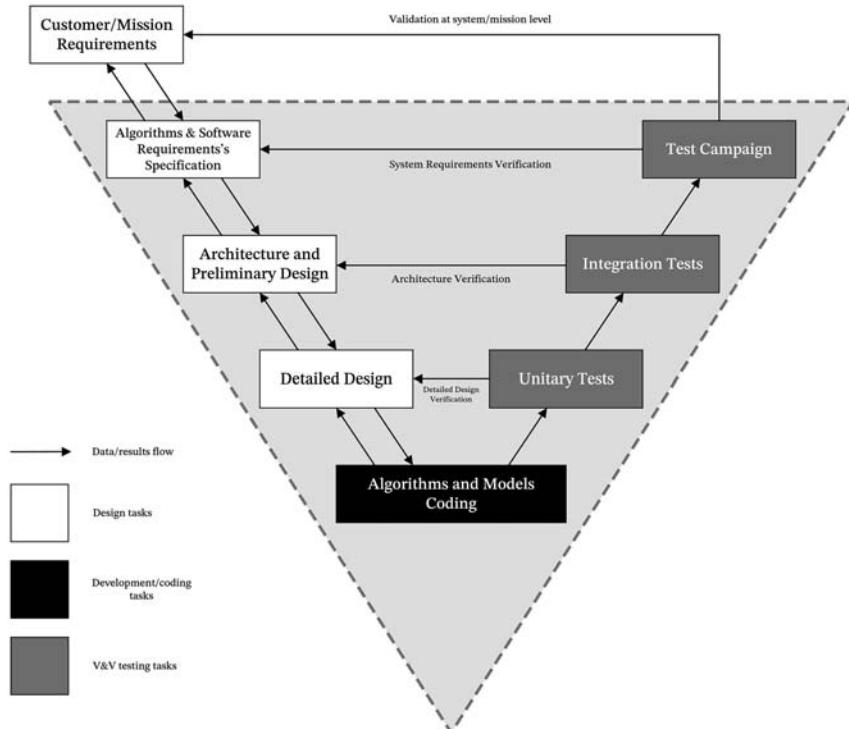
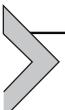


Figure 1.8 Model-based design and prototyping.

unitary tests that determine whether the algorithms and functions behave properly as stand-alone functions. Next, the software is (partially) integrated and integration tests are performed to ensure that all the components of the software work well together. Finally, a full test campaign is performed to ensure that the software meets all the performance specifications that have been established in the requirements documents.

Fig. 1.8 shows a single waterfall design cycle, but in an actual design process, more iterations and customer feedback take place at each step of this process. The feedback can result in modifications of any part of the requirements and the software design. At the same time, early implementation and testing may occur to help in the definition of the requirements. Finally, design projects at different stages of the mission life cycle can make different loops through the waterfall that serve to consolidate the user requirements for the next phase of the project. In this case, the set of user requirements for the next phase of the design can come with a preliminary design that was established during the phase that came before.



Notation rules

The main notation rules used in this book are detailed in this paragraph.

Vectors and matrices are represented with bold symbols or in parenthesis as follows:

$$\mathbf{x}_{nx1} = \{x_i\} = \begin{Bmatrix} x_1 \\ \vdots \\ x_n \end{Bmatrix}, \mathbf{A}_{nmx} = [A_{ij}] = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ \vdots & \vdots & & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{bmatrix}.$$

Most of the times, vectors are indicated with small letter symbols, while matrices with capital letter symbols. Therefore, the previous notation indicates column vectors of n components and matrices of n rows and m columns. Note that the subscript with the dimensions will be indicated only if necessary. The single component is represented by the nonbold symbol with one or two indices. A row vector is always the transpose of a column vector:

$$\mathbf{x}^T = \{x_i \cdots x_n\}.$$

The inverse of a nonsingular square matrix is represented by the symbol $(\)^{-1}$, and it is defined by the condition:

$$\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n, \text{ with } I_{ij} = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{if } i \neq j \end{cases}$$

where \mathbf{I}_n is the identity matrix with size n .

It is assumed that the reader is familiar with matrix algebra, also briefly summarized in [Appendix Chapter 16](#) – Mathematical and Geometrical Rules. In this book, the dot product will be represented with the dot symbol “.” and the vector product with the cross symbol “ \times ”. The dot symbol “.” always refers to a dot product of two vectors; on the contrary, the product of two scalars x and y will be written as xy , with no specific symbol.

The cross product operation is also commonly indicated in matrix form, by exploiting the cross product matrix that is defined for a generic vector in three-dimensional space as:

$$[\mathbf{x}_{3x1} \times] = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix},$$

which is skew-symmetric (i.e., its transpose is equal to the same matrix with opposite sign).

A general quaternion will be expressed as $\mathbf{q} = q_1 i + q_2 j + q_3 k + q_4$. Here i, j, k are the units of quaternions, satisfying $i^2 = j^2 = k^2 = ijk = -1$ and q_1, q_2, q_3, q_4 are all real numbers. q_4 is the scalar part of the quaternion, and $q_1 i + q_2 j + q_3 k$ is the vector part of the quaternion.

Notation table

A more comprehensive list of the used notation symbols is reported in the table:

Vectors

$\mathbf{x}_{nx1} = \{x_i\} = \begin{Bmatrix} x_1 \\ \vdots \\ x_n \end{Bmatrix}$	Column vector with n entries $x_1 \dots x_n$
$x_{r:s}$	Subvector with entries from r to s
$x = \ x\ $	Norm of vector x
$\hat{\mathbf{x}}_{nx1} = \frac{\mathbf{x}_{nx1}}{\ \mathbf{x}_{nx1}\ }$	Unit norm vector with n entries
$\text{rms}(x)$	Root-mean-square value of vector x
$\text{avg}(x)$	Average of entries of vector x
$\text{std}(x)$	Standard deviation of vector x

Matrices

$\mathbf{A}_{n xm} = [A_{ij}] = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ \vdots & \vdots & & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{bmatrix}$	$n \times m$ matrix with entries $A_{11} \dots A_{mn}$
$\mathbf{A}_{r:s, p:q}$	Submatrix with rows r to s and columns p to q
\mathbf{I}_n	Identity matrix with size n
\mathbf{A}^T	Transpose of matrix \mathbf{A}
$\ \mathbf{A}\ $	Norm of matrix \mathbf{A}
\mathbf{A}^k	(Square) matrix \mathbf{A} to the k th power
\mathbf{A}^{-1}	Inverse of matrix \mathbf{A}
$\text{diag}(\mathbf{x})$	Diagonal matrix with diagonal entries $x_1 \dots x_n$
$\text{diag}(\mathbf{A})$	Diagonal entries $x_1 \dots x_n$ of the matrix \mathbf{A}
$[\mathbf{x}_{3x1}] = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$	3×3 skew-symmetric cross-product matrix

—cont'd

Quaternions

$\mathbf{q} = q_1 i + q_2 j + q_3 k + q_4$	Quaternion vector, scalar part last
$\bar{\mathbf{q}} \times \mathbf{q} = \begin{bmatrix} q_4 \bar{\mathbf{q}}_{1:3} + \bar{q}_4 \mathbf{q}_{1:3} - \bar{\mathbf{q}}_{1:3} \times \mathbf{q}_{1:3} \\ \bar{q}_4 q_4 - \bar{\mathbf{q}}_{1:3} \cdot \mathbf{q}_{1:3} \end{bmatrix}$	First added quaternion operation
$\mathbf{q}^* = -q_1 i - q_2 j - q_3 k + q_4$	Conjugate quaternion
$\mathbf{q}^{-1} = \mathbf{q}^*/\ \mathbf{q} \ $	Inverse quaternion

Functions and derivatives

$f : A \rightarrow B$	f is a function on the set A into the set B
∇f	Gradient of function $f : \mathbf{R}^n \rightarrow \mathbf{R}$ at z
$\frac{df}{dx} = \left\{ \frac{\partial f}{\partial x_i} \right\}$	Derivative of function f with respect to vector \mathbf{x}
$\nabla_{\mathbf{x}} f$	Jacobian of vectorial function $f : \mathbf{R}^n \rightarrow \mathbf{R}^m$
$\nabla_{\mathbf{x}}^2 f$	Hessian matrix of f
$\dot{f} = \frac{df}{dt}$	Time derivative of function f
\bar{f}	Reference function
$\delta f = f - \bar{f}$	Perturbation function

Random variables

$E\{\mathbf{x}\} = \boldsymbol{\mu}$	Expected values of a random vector
$\mathbf{P}\{\mathbf{x}\} = \boldsymbol{\Sigma}\{\mathbf{x}\} = E\{(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\}$	Covariance matrix
$\sigma^2\{\mathbf{x}\} = \text{diag}(\mathbf{P}) = E\{(\mathbf{x} - \boldsymbol{\mu})^2\}$	Variance
$\sigma\{\mathbf{x}\} = \sqrt{\sigma^2\{\mathbf{x}\}}$	Standard deviation

Errors and variables elaboration

$\mathbf{x}(t)$	True variable
$\hat{\mathbf{x}}(t)$	Measured variable
$\tilde{\mathbf{x}}(t)$ or $\widehat{\mathbf{x}}(t)$	Estimated variable
$\bar{\mathbf{x}}(t)$ or $\mathbf{x}_{\text{ref}}(t)$	Reference variable
$\delta \mathbf{x} = \mathbf{x}(t) - \bar{\mathbf{x}}(t)$	Error variable
$\delta \mathbf{q} = \mathbf{q} \times \bar{\mathbf{q}}^{-1}$	Error quaternion

List of Acronyms

Artificial Intelligence	AI
Artificial Neural Networks	ANNs
Assembly Integration and Test	AIT
Assembly Integration and Verification	AIIV

Attitude and Orbit Control System	AOCS
Attitude Determination and Control System	ADCS
Automated Code Generation	ACG
Center of Mass	CoM
Central Processing Unit	CPU
Circular Restricted Three-Body Problem	CRTBP—CR3BP
Collision Avoidance Maneuver	CAM
Commercial Off the Shelf	COTS
Control Moment Gyros	CMG
Deep Learning	DL
Degrees of Freedom	DoF
Digital Signal Processors	DSPs
Electrical Ground Support Equipment	EGSE
Elliptic Restricted Three-Body Problem	ERTBP—CR3BP
European Cooperation for Space Standardization	ECSS
Extended Kalman Filter	EKF
Failure Detection Isolation and Recovery	FDIR
Failure Mode Effects and Criticality Analysis	FMEA
Fault Tree Analysis	FTA
Field Programmable Gate Array	FPGA
Geostationary Orbits	GEOs
Global Navigation Satellite System	GNSS
Graphical Processing Units	GPUs
Ground Support Equipment	GSE
Guidance, Navigation, and Control	GNC
H-Infinity	H^∞
Hardware	HW
Hardware-in-the-Loop	HIL
In-Orbit testing	IOT
Inertial Measurement Unit	IMU
Inertial Navigation System	INS
International Geomagnetic Reference Field	IGRF
Kalman Filter	KF
Kee Out Zone	KOZ
Launch and Early Orbit Phase	LEOP
Light Detection and Ranging	LIDAR
Linear Quadratic Regulator	LQR
Low Earth Orbit	LEO
Machine Learning	ML
Medium Earth Orbits	MEOs
Minimum Impulse Bit	MIB
Mission and Vehicle Management	MVM
Model Predictive Control	MPC
Model-in-the-Loop	MIL
On-Board Software	OBSW
Particle Filter	PF
Processor-in-the-Loop	PIL
Proportional-Integral-Derivative	PID
Reinforcement Learning	RL
Reliability, Availability, Maintainability, and Safety	RAMS

Singular Value Decomposition	SVD
Sliding Mode Control	SMC
Software	SW
Software-in-the-Loop	SIL
Solar Radiation Pressure	SRP
Special Check-Out Equipment	SCOE
Sun-Synchronous Orbits	SSOs
Technology Readiness Level	TRL
Two-Line Elements	TLE
Unscented Kalman	UKF
Validation and Verification	V&V

References

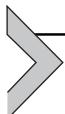
- [1] J.B. Hammack, Postlaunch Report for Mercury-Redstone No. 3 (MR-3), NASA Report, 1961.
- [2] ECSS-E-ST-60-30C, Satellite Attitude and Orbit Control System (AOCS) Requirements, ECSS Standards, 2013.
- [3] ECSS-S-ST-00-01C, Glossary of Terms, ECSS Standards, 2012.
- [4] F.L. Markley, J.L. Crassidis, Fundamentals of Spacecraft Attitude Determination and Control, Space Technology Library, Springer, New York, 2014.
- [5] R.J. Duxbury, Air Force Space Command, Space Vehicle Failure Modes, Effects, and Criticality Analysis (FMECA) Guide, Space Missile System Center, 2009. El Segundo, CA, USA, Aerosp. Rep. No. TOR-2009 (8591)-13.
- [6] ECSS-Q-ST-30-02C — Failure Modes, Effects (And Criticality) Analysis (FMEA/ FMECA), ECSS Standards, 2009.
- [7] M. Jackson, C. D'Souza, H. Lane, Autonomous Mission Management for Spacecraft Rendezvous using an Agent Hierarchy, in: Infotech@ Aerospace, 2005, p. 7062.
- [8] R. Turner, et al., ExecSpec: visually designing and operating a finite state machine-based spacecraft autonomy system, in: 9th International Symposium on Artificial Intelligence, Robotics and Automation for Space (i-Sairas'08), Pasadena, CA, 2008.
- [9] W.M. Johnston, et al., Advances in dataflow programming languages, ACM Computing Surveys 36 (1) (March 2004) 1–34.
- [10] <https://www.mathworks.com/products/stateflow.html>.
- [11] W. Truszkowski, et al., Autonomous and Autonomic Systems: With Applications to NASA Intelligent Spacecraft Operations and Exploration Systems, Springer Science & Business Media, 2009.
- [12] D.K. Geller, Orbital rendezvous: when is autonomy required? Journal of Guidance, Control, and Dynamics 30 (4) (2007) 974–981.
- [13] J.R. Wertz, W.J. Larson, in: Space Mission Analysis and Design, Microcosm, 1999.
- [14] <https://www.h2020-ergo.eu/project/background-on-autonomy-software-frameworks-autonomy-in-space-systems/>.
- [15] P. Grandjean, T. Pesquet, A.M.M. Muxi, M.C. Charneau, What on-board autonomy means for ground operations: an autonomy demonstrator conceptual design, in: Space OPS 2004 Conference, 2004, p. 267.
- [16] R. Sterritt, D.W. Bustard, Autonomic computing—a means of achieving dependability?, in: Proc. IEEE International Conference on the Engineering of Computer Based Systems (ECBS-03), Pages 247–251, Huntsville, Alabama (USA) IEEE Computer Society Press, Los Alamitos, California (USA), April 2003.



PART ONE

Fundamental GNC tools

This page intentionally left blank



Reference systems and planetary models

Andrea Colagrossi¹, Stefano Silvestrini¹, Vincenzo Pesce²

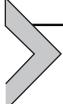
¹Politecnico di Milano, Milan, Italy

²Airbus D&S Advanced Studies, Toulouse, France

This chapter presents an overview of the key concepts regarding generic planetary models and the definition of standardized time models, as well as relevant reference systems commonly used in the Guidance Navigation and Control (GNC) design. The notions of this chapter, although here only briefly outlined, are extremely important to avoid inherent and intrinsic errors in the system design. Also, a careful analysis of the presented topics prevents ambiguities when dealing with typical GNC operations, scheduling, and, in general, system and algorithm features that require unique and well-established representations of the environment.

The content of this chapter is structured as follows:

- *Earth and planetary models.* This section describes the models to describe planetary geometry, with focus on the Earth's ellipsoid and geoid. It also includes a brief introduction to position representation methods for objects on and above the planetary surface.
- *Coordinate reference systems.* In this section, the most relevant coordinate reference systems are presented. Heliocentric, geocentric, topocentric, lunar, three-body, and satellite-based reference systems are discussed.
- *Coordinate transformations.* This section briefly explains the methods to transform the coordinates from one reference system to another one, with some examples of the most relevant transformations for GNC applications.
- *Time.* The most relevant time conventions and scales are discussed in this section. The most common time systems are described, and the concept of Julian dates (JDs) is introduced.
- *What is relevant for GNC?* This section summarizes and highlights the most relevant concepts for GNC applications discussed in this chapter.



Earth and planetary models

The description of the Earth or the planet around which a spacecraft is orbiting is a preliminary task every GNC designer must accomplish before tackling the GNC system design. In fact, we must define a set of fundamental parameters to specify the shape and the geometry of any planet. These values allow us to specify locations, shape, the planet's precise size, and the gravity field. This task is known as geodesy. We will focus the following discussion on the Earth, but it can be easily generalized and extended to most of the planets and natural satellites in the Solar System, as long as the proper reference models are found and defined according to the international standards.

Earth and most planetary shapes can be conveniently described exploiting an oblate ellipsoid (i.e., an ellipsoid resulting from the revolution of an ellipse around its minor axis), based on two physical characteristics:

- The equatorial radius, R_E .
- The planetary eccentricity or flattening, e_E or f_E .

The Earth's radius has been investigated since the beginning of human presence, and the first calculation made by Eratosthenes in the third century BC was astonishingly accurate (i.e., he made an error in the order of 1%–2% with respect to the current values). Modern efforts began to converge on an accepted value in the mid-1800s. The WGS-84 model [1] defines the mean equatorial radius of the Earth, R_E , as:

$$R_E = 6\,378\,137 \text{ m}$$

Eccentricity and flattening of the Earth's ellipsoid are in WGS-84:

$$e_E = 0.081\,819\,190\,842$$

$$f_E = 0.003\,352\,810\,664 = \frac{1}{298.257\,223\,563}.$$

We can also calculate the semiminor axis of the Earth's ellipsoid, using R_E as the semimajor axis, to aid in formulas relating positions on the Earth's surface. The semiminor axis, b_E , also called the polar axis, is in WGS-84:

$$b_E \cong 6\,356\,752.\overline{314\,245} \text{ m}$$

Note that this is a derived quantity, and the overline bar indicates the digits beyond the original accuracy in R_E . The equatorial radius and polar axis are related with the flattening as: $f_E = (R_E - b_E)/R_E$. The flattening is also denoted as oblateness.

The WGS-84, World Geodetic Survey 1984 latest version, is a standard geodetic model for use in geodesy and satellite navigation, including global navigation satellite system (GNSS). This standard includes the definition of the coordinate system's fundamental and derived constants, the normal gravity Earth Gravitational Models (EGMs) (EGM-96/EGM-2008 [2,3]), and it is the reference ellipsoid for the standard Internal Geomagnetic Field Model (IGRF). WGS-84 is of primary use and definition by the American associations, while European standards [4] suggest the use of EIGEN-GL04C as a standard for gravity field modeling and geodesy. Note that European standards require anyway the use of WGS-84 as a reference ellipsoid for IGRF usage. In any case, EIGEN-GL04C defines the Earth ellipsoid with an equatorial radius of $R_E = 6\,378\,136$ m, a polar radius of $b_E = 6\,356\,752$ m, and an oblateness of $f_E = 1/298.257$. Hence, the differences with respect to WGS-84 are minimal for what concern the geodesy and the reference ellipsoid definition.

Additional physical characteristics of the Earth are:

- The rotational velocity, ω_E .
- The gravitational parameter, μ_E .

The Earth's rotational velocity, ω_E , is frequently assumed to be constant in time, and indeed it seemed to be so for many years, given the limitation of existing measurement instruments. The accepted constant value for the Earth's rotation is:

$$\omega_E = 7.292\,115\,\overline{146\,706\,980} \times 10^{-5} \text{ rad/s.}$$

The last parameter is the standard gravitation parameter, which is now commonly measured from satellite observations, rather than traditional methods (e.g., Cavendish method [5]). Although the precise definition includes the masses of the Earth and the satellite, we neglect the satellite's mass because it's so small relative to the Earth's. Then, the standard gravitation parameter of the Earth is:

$$\mu_e = G(m_E + m_{s/c}) \cong Gm_E = 3.986\,004\,415 \times 10^5 \text{ km}^3/\text{s}^2,$$

where $G = 6.673 \times 10^{-20} \text{ km}^3/(\text{kg s}^2)$ is the gravitational constant, and m_E is the mass of the Earth. The value of μ_e is the same in EGM-96, EGM-08 and EIGEN-GL04C. Note that, μ_e and G are experimentally measured, while m_E is a derived quantity. The standard gravitational parameters of other celestial bodies and extensive discussions about several Earth and planetary models may be found in Ref. [6].

Position representation

To represent the position of a spacecraft in space, cartesian coordinates may be used. They are typically defined with respect to inertial, planetary-fixed, or orbit-fixed three-dimensional cartesian reference frames. The most useful for GNC application will be described in the next section of this chapter. An alternative and very convenient approach to represent the position of a body around the Earth, or a planet, is to use spherical coordinates, defining a radius vector and two angles, which are typically denoted as latitude, or declination, and longitude, or right ascension. The latter terms (i.e., declination and right ascension) are typically used to refer to position with respect to the celestial sphere and inertial reference frames, the former (i.e., latitude and longitude) are more commonly used around one planet, as seen in Fig. 2.1. However, it is important to note that it is only a terminological difference, depending from the reference directions from which the angles are measured. Mathematically, they are analogous quantities: the two angles used in spherical coordinate frames.

Focusing on position representation on the Earth, latitude and longitude are the common spherical angles to define the position of a spacecraft with

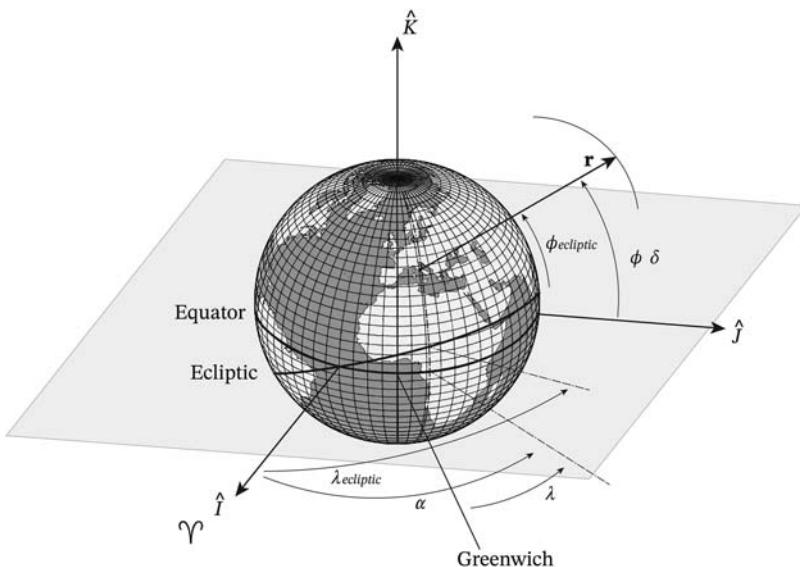


Figure 2.1 Right ascension, α , declination, δ , longitude, λ , latitude, φ , as spherical coordinates to define position. They are typically referred to the equatorial plane, but they can be referred to other fundamental planes. For example, ecliptic longitude, $\lambda_{\text{ecliptic}}$, and ecliptic latitude, $\varphi_{\text{ecliptic}}$, shown in the figure.

respect to the Earth's surface. Latitude is the north–south angular measurement with respect to the equatorial plane (i.e., the plane containing the Earth's equator). It takes on values from 0° to $\pm 90^\circ$, and it's positive in the northern hemisphere. Longitude is an east–west angular displacement measured positive to the east from the plane containing the prime meridian. The prime meridian for the Earth is the meridian (i.e., the intersection of a plane passing through the Earth's axis and the Earth's surface) that the Royal Observatory at Greenwich lies on. Longitude may take on values from 0° to 360° when measured east, or from 0° to $\pm 180^\circ$ if measured east—positive—and west.

However, the Earth and most of the celestial bodies are not perfect spheres. Thus, to precisely locate an object with respect to their surface, latitude and longitude definitions may become a bit more complicated, also depending on the planet of interest. For example, the Moon is better represented by a triaxial ellipsoid (i.e., an ellipsoid with three different semiaxes). While, as already discussed, the Earth can be represented by a simpler oblate ellipsoid, whose semimajor axis is equal to the equatorial radius, R_E , and semiminor axis is equal to the polar radius, b_E . In this case, longitude, λ , has no definition ambiguities, due to the ellipsoid rotation around the polar axis, while latitude shall be defined considering the flattening of the Earth. With references to Fig. 2.2, we can define:

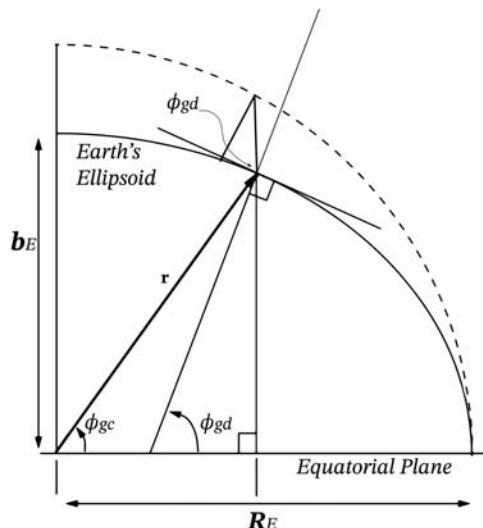


Figure 2.2 Geocentric and Geodetic Latitude: The geodetic latitude, φ_{gd} , makes an angle perpendicular to the surface and the equatorial plane, whereas the geocentric latitude, φ_{gc} , is referenced to the center of the Earth.

- Geocentric latitude, φ_{gc} , as the angle measured at the Earth's center from the plane of the equator to the point of interest. Geocentric latitude is equal to the spherical latitude definition.
- Geodetic latitude, φ_{gd} , as the angle between the equatorial plane and the normal to the surface of the ellipsoid. The latitude on most maps is geodetic latitude.

To convert between geocentric latitude-longitude values and cartesian coordinates, simple geometrical considerations are necessary. A position with respect to the Earth can be defined as:

$$\mathbf{r} = \begin{bmatrix} r_I \\ r_J \\ r_K \end{bmatrix} = \begin{bmatrix} r \cos(\varphi_{gc}) \cos(\lambda) \\ r \cos(\varphi_{gc}) \sin(\lambda) \\ r \sin(\varphi_{gc}) \end{bmatrix}.$$

For locations on the Earth's surface (i.e., r corresponding to the ellipsoidal radius), we can define a conversion between geocentric and geodetic latitude as:

$$\tan(\varphi_{gd}) = \frac{\tan(\varphi_{gc})}{1 - e_E^2}$$

Note that the above equation can be inverted to compute φ_{gc} from a given φ_{gd} , but it is valid only for locations on the Earth's ellipsoid (i.e., on the planetary surface). Neglecting the difference between geocentric and geodetic latitudes can cause errors up to about 20 km.

Above the Earth's surface, like for orbiting spacecraft, we shall first refer to declination, δ , and right ascension, α , with respect to the inertial celestial sphere. Then, we shall convert to the Earth fixed position to find geodetic latitude and longitude of the spacecraft's sublatitude point. The sublatitude point on the Earth's surface is defined as the point which lies directly perpendicular below the satellite, and it is generally different from the surface point which lies along the position vector, as represented in Fig. 2.3. When sublatitude point of the spacecraft is found, we can convert between geocentric and geodetic latitudes of sublatitude point using the previous equation valid on the planetary surface. In general, we shall note that geocentric latitude of the spacecraft, $\varphi_{gc_{sat}}$, is conceptually equivalent to declination, but geocentric and geodetic latitudes of sublatitude point on the Earth's surface are different from declination of the satellite.

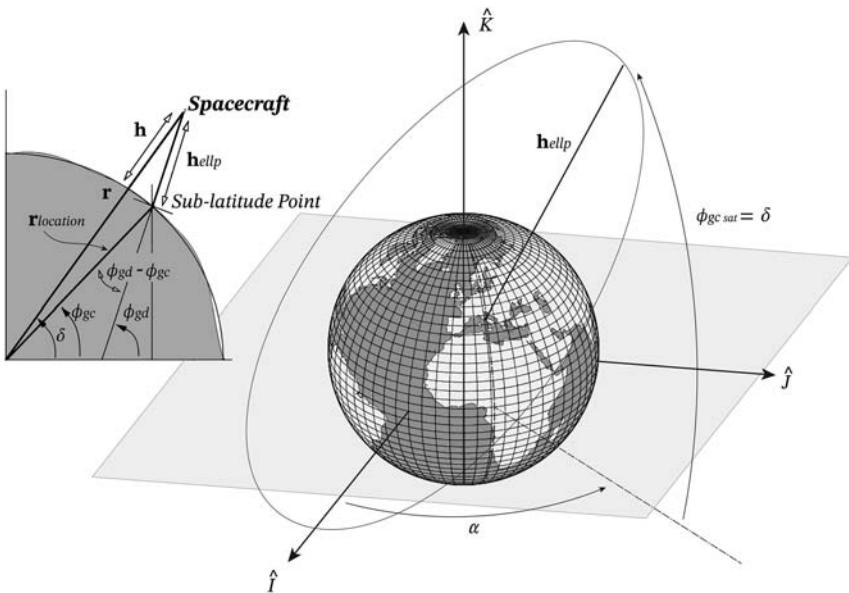


Figure 2.3 Spherical coordinates above ground and satellite sublatitude point.

Converting the position vector of a satellite to the corresponding latitude and longitude is the core technique in determining ground tracks and in designing GNC applications with respect to the Earth's (or planet's) surface. First, we need to remark that if the planet were a perfect sphere, the algebraic relationship between spherical and cartesian coordinates would have a direct solution. With ellipsoidal and more complex shapes, there exists two ways to perform the transformation of position to latitude and longitude: one iterative and one analytical.

Both methods are extensively described, also with pseudocode examples in Ref. [6]. Here, we just want to highlight the main steps of the procedure.

The right ascension can be directly computed from the cartesian position vector as:

$$\sin(\alpha) = \frac{r_J}{\sqrt{r_I^2 + r_J^2}} \quad \text{and} \quad \cos(\alpha) = \frac{r_I}{\sqrt{r_I^2 + r_J^2}},$$

and the approximate conversion to longitude is basically an angular shift with respect to the prime meridian reference:

$$\lambda \cong \alpha - \theta_{GMT},$$

where θ_{GMT} is the right ascension of Greenwich mean meridian.

This is only an approximate formula to explain the fundamental idea behind this conversion, while the complete conversion is discussed in the following, when the transformation between Earth-centered inertial (ECI) and Earth-centered, Earth-fixed (ECEF) frames is presented. The next steps to find the geodetic latitude are the most difficult, where the iterative approach or the analytical one is used. To determine a starting value for the iterations, we can use the position vector as a rough guess because the declination and geocentric latitude of the satellite are equal:

$$\sin(\delta) = \frac{r_K}{r}.$$

At this point, the geodetic latitude can be computed with one of the dedicated methods described in the cited reference [6].

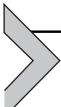
Geoid and geopotential models

The reference ellipsoid is a good approximation of the hypothetical surface referred to as mean sea level (MSL). The actual MSL surface, excluding small atmospheric effects, is called the geoid, and it deviates from the reference ellipsoid because of the uneven distribution of mass in the Earth's interior. The geoid is a geopotential surface—a plumb bob will hang perpendicular to it at every point—because the gravity potential is equal at all points and the direction of gravity is always perpendicular to the geoid surface. Hence, the concept of height relies on geopotential surfaces at which the gravity is equal at all points. The geoid's undulation, N_E , is the geoid's height above the ellipsoid, whereas the actual height of the topography above the geoid is called the orthometric height, H_{MSL} . The latter is the height we all used because of maps and road signs. Finally, the ellipsoidal height, h_{ellp} , is like the orthometric height, but it's measured perpendicular to the ellipsoid and differs by the geoid undulation:

$$N_E \cong h_{ellp} - H_{MSL}.$$

The geoid definition is closely related to the gravity potential expressed with spherical harmonics expansions, which will be extensively described in [Chapter 3](#)—The Space Environment. In fact, the geoid's undulation can be mathematically represented exploiting similar harmonics expansions. However, geoid's undulation takes on values between +85 and −107 m, and the full series expansions must be used only for very precise applications. In most GNC utilizations, N_E can be tabulated in a grid of $1^\circ \times 1^\circ$ or even $10^\circ \times 10^\circ$. Locations not directly contained on the grid are found using interpolation techniques and two-dimensional look-up tables. Note that

the geoid geopotential model should be used in agreement with the used gravitational model. For instance, EIGEN-GL04C geoid shall be used together with EIGEN-GL04C gravitational coefficients, while WGS-84 geoid shall be used together with the EGM coefficients (e.g., EGM-96 or EGM-08).



Coordinate reference systems

When a GNC system has to be designed and analyzed, one of the first decision to take is the definition of a suitable reference system or a set of them. We define a rectangular coordinate system by specifying its origin, fundamental plane, and preferred positive directions. For most reference systems, the preferred positive directions have a right-handed sense. We use three unit vectors to represent three orthogonal axes and express any other vector in the coordinate system as a linear combination of the base unit vectors. We have to bear in mind that, despite the definition in rectangular cartesian coordinates, the spherical coordinates may be obtained by converting the vector components as described above. This can be helpful for some applications where two angles and a radius vector are more explicative than three cartesian components.

Heliocentric coordinate system, XYZ

For GNC applications involving interplanetary missions, a coordinate reference system centered in the Sun is very useful. The Heliocentric Coordinate System, *XYZ*, has its origin at the center of the Sun, and the fundamental plane is the ecliptic. The primary direction for this system, *X*, is the intersection of the ecliptic and Earth's equatorial planes. The positive direction of the *X* axis is given by the location of the Sun with respect to the Earth as it crosses the equatorial plane on the first day of spring: vernal equinox. In other words, the *vernal equinox*, or first point of Aries, is the direction of the line that joins the Earth with the Sun when the Sun's declination is 0° and it changes from negative to positive. The *Z* axis is orthogonal to the ecliptic plane, positive in the direction of the Earth's angular momentum. The *Y* axis completes the right-handed triad. The location of an object in this reference frame can be also expressed in terms of ecliptic latitude and longitude.

In some cases, the center of the interplanetary reference system is at the barycenter of the Solar System, and not at the center of the Sun, as in the International Celestial Reference Frame (*ICRF*), which is the current

standard International Astronomical Union (IAU) inertial reference system. The ICRF is defined through observations of compact extragalactic radio sources. These astronomical objects are so far away that their expected motions should be negligibly small. The precise definition of an interplanetary reference system may be subject to periodic reevaluation after accurate observations are performed to update the location of the reference astronomical objects. In any case, the new solutions introduce no rotation from previous realizations, since numerical operations benefit from the axes remaining fixed.

Geocentric equatorial coordinate system, IJK (ECI)

This system originates at the center of the Earth. The fundamental plane is the Earth's equator. The I axis points toward the vernal equinox, and the K axis extends through the North Pole. J completes the right-handed triad. The geocentric frame can be also referred as ECI, and it is reported in Fig. 2.4.

Since both the equinox and the equatorial plane slightly move over time, due to precession, nutation, and other secular motions, the above definition does not represent an inertial frame. A “pseudo” Newtonian inertial system can be obtained by referring to the axes directions at a particular epoch, specifying the transformations to move from the current epoch to the reference one and vice versa. Throughout the years, different reference systems at epoch have been proposed:

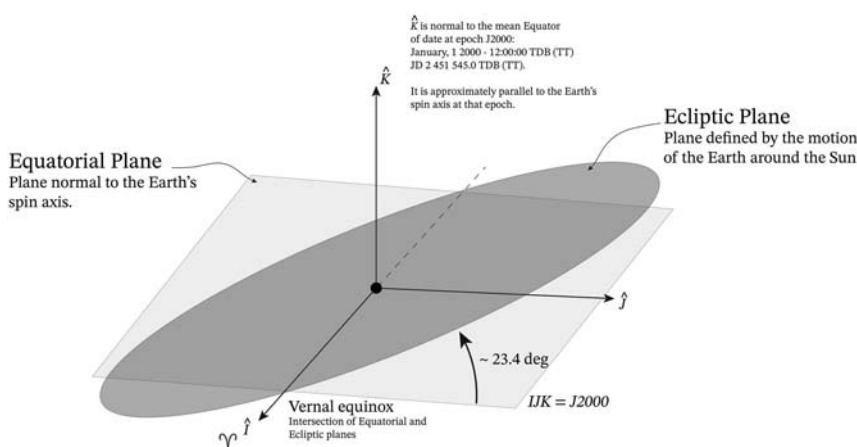


Figure 2.4 Example of ECI. J2000 reference system.

J2000 is an inertial frame realized in the IAU-76/FK5 system, which is based on the *Fundamental Katalog*, FK5 star catalog. For many years, it has been the standard reference system for geocentric coordinates. The system exploits the equator, equinox, and polar axis directions at 12:00 (i.e., noon) TDB (Barycentric Dynamical Time), on January 1st, 2000. Numerical data to transform other systems to the J2000 were obtained from the IAU-1976 Precession Model and the IAU-1980 Theory of Nutation.

Geocentric Celestial Reference Frame (GCRF) is the geocentric counterpart of the ICRF, and it is the current standard inertial coordinate system for the Earth. To provide continuity with former references, the axes directions are selected to be as close as possible to the J2000 frame. Note that only a very small rotational difference (i.e., <0.1 arcseconds) exists between the two frames. It was officially adopted in 1997, effectively replacing the IAU-76/FK5 J2000 system. As the ICRF, the GCRF is subject to periodic reevaluations which increase the stability of the axes by adding more defining sources and improving their coordinates. Nevertheless, the newest versions do not introduce any rotations with respect to previous ones. The International Earth Rotation and Reference Systems Service (IERS) has maintained tabulated corrections to transform a vector from the GCRF to the IAU-76/FK5 J2000 frame.

These pseudoinertial systems are considered as inertial for precise orbit determination and calculations. In this book, the acronym ECI will be used interchangeably with the GCRF.

ECI spherical coordinates are commonly denoted as:

- *Right ascension*, α : the angle measured eastward on the plane of the equator from the vernal equinox to the celestial meridian that contains the object.
- *Declination*, δ : the angle between the equatorial plane and the object, measured in the meridional plane that passes through the object, positive above the equator.
- *Radial distance*: the distance between the Earth's center and the object.

Geocentric earth-fixed coordinate system, $I_FJ_FK_F$

A geocentric coordinate system fixed to the rotating Earth results in the ECEF coordinate frame. The ECEF is a noninertial geocentric coordinate system that rotates with the Earth, and it is fixed with respect to the Earth's surface. The official IAU earth-fixed terrestrial frame is named the International Terrestrial Reference Frame (*ITRF*). The ITRF origin is at the center

of the Earth, and the axes have been defined through the coordinates of a set of stations on the Earth's surface. Since the tectonic plate motion affects the position of these reference locations, the ITRF is regularly updated such that there is no net rotation with respect to previous versions. In this book, the acronym ECEF will be preferably used for referring to the ITRF. In the ITRF, the Z axis is parallel to the direction of the North Pole, the X axis is defined as the intersection between the fundamental plane (perpendicular to Z) and the Greenwich mean meridian. The Y axis is orthogonal to X and Y . ECEF frame is reported in Fig. 2.5.

As this coordinate system rotates, the epoch when observation and data are acquired must be specified to allow the transformation to the ECI/GCRF system. The time precision to perform this transformation is crucial to maintain the accuracy of the measurements without introducing rotation-related errors. The ECEF is based on the Earth's equatorial plane. It is especially useful to process satellite observations to/from a specific site and to convert Earth-based data to ECI for further processing.

ECEF spherical coordinates are differently denoted with respect to the ECI as:

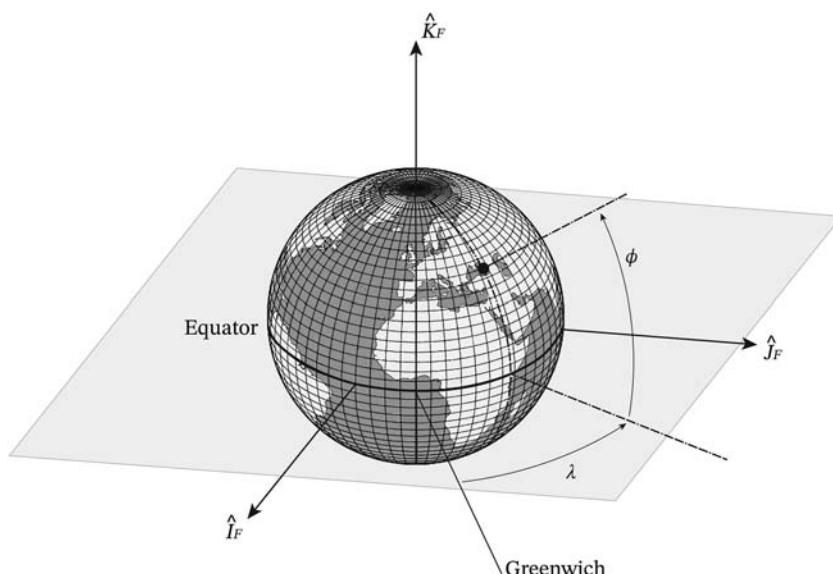


Figure 2.5 Example of ECEF reference system. λ and φ are the east longitude and geocentric latitude, respectively.

- *East longitude*, λ . The angle measure eastward in the equatorial plane, from the Greenwich meridian to the meridian containing the object.
- *Geocentric latitude*, φ_{gc} . The angle between the object and the equatorial plane, measured in the meridional plane that passes through the object, positive above the equator.

A similar reference frame exists, and it is based on the WGS-84 model. It is practically identical to the ITRF; however, it has been defined through Global Positioning System (GPS) measurements. The WGS-84 and the ITRF differ only at cm level. Within the uncertainty of the WGS-84 frame, they are equivalent.

Orbit determination and many GNC applications typically require both celestial inertial reference frames and terrestrial (or planetary) reference frames. The former defines the Newtonian-inertial space in which differential equations of satellite motion are valid, while the latter is commonly used to take some relevant observations and measurements.

Topocentric coordinate systems

Topocentric coordinate systems are centered at a site on the Earth's surface. So, they are Earth-based systems, but they are not geocentric. Different topocentric frames exist; a brief survey is here reported.

Topocentric equatorial

This system is equal to the geocentric equatorial coordinate system (ECI), but it has the origin translated from the Earth's center toward the observer location on the Earth's surface. The directions of the axes are parallel to the ECI reference frames. The location of an object is then identified through its *topocentric right ascension* and *declination*. However, since the geocentric and topocentric systems do not have the same origin, for an earth-orbiting satellite, the topocentric right ascension and declination will be different from the geocentric ones. On the other hand, if the object is far away from Earth (e.g., distant planets or stars), the difference between the two position vectors becomes negligible. It is mainly used for highly accurate optical systems.

Topocentric horizon

In a topocentric horizon coordinate system, the fundamental plane is defined by the local horizon (i.e., the plane tangent to the ellipsoid at the observer location), and the reference is centered at the observer site on the Earth's

surface. If the observer is not located on the Earth's surface (e.g., an aircraft), the fundamental plane is still oriented along the geodetic directions defined by the Earth's surface below it. Topocentric horizon coordinate systems are very useful in observing satellites from ground, and they are used extensively with Earth-based sensor systems. Depending on which directions the reference axes are pointing, different topocentric horizon frames exist:

- **SEZ (South, East, Zenith).** The X axis points south, even in the southern hemisphere. The Y axis is directed east from the site and is undefined for the North or South Pole. The Z axis points radially outward from the site, along the site's local vertical (i.e., the Zenith). Fig. 2.6 reports an example of SEZ reference frame.
- **NED (North, East, Down).** The X axis points north, the Y axis is directed eastward, and the Z axis points inwards, toward the nadir. This system is particularly useful on-board of airplanes, as most objects of interest are below the aircraft.

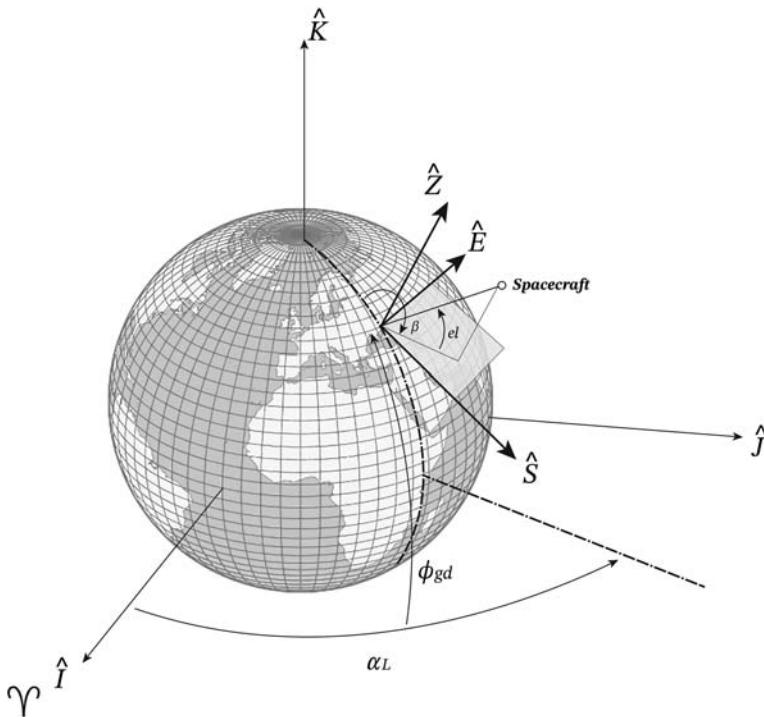


Figure 2.6 Topocentric horizon reference system, SEZ.

- *ENZ (East, North, Zenith)*. The X axis points east, the Y axis is directed north, and the Z axis points outwards toward the Zenith. It is obtained by rotating the SEZ reference frame of 90° around the Z axis.

Note that the observer location on the Earth is defined as exploiting geodetic latitude and right ascension of the site, α_L . Alternatively, east longitude of the site may be used, but the right ascension of the prime meridian must be known.

Topocentric reference spherical coordinates are denoted as:

- *Azimuth, β* . The angle measured from north, clockwise (as viewed from above) to the location beneath the object of interest.
- *Elevation, el* . The angle measured from the local horizon, positive up to the object of interest. It takes on values from -90° to 90° . We rarely encounter negative values for sites on the Earth, but they often appear with an orbiting satellite.

Lunar coordinate systems

There is not an official definition of a Lunar-Centered Inertial (LCI) reference frame; however, such system can be straightforwardly obtained by translating the origin of the GCRF toward the Moon center. On the other hand, different lunar-fixed reference systems have been proposed.

Mean earth/polar axis

The Mean Earth/Polar Axis (ME) reference system is a lunar body-fixed coordinate system. It exploits the mean direction of Earth to define the prime meridian (i.e., 0° in longitude). In some reference, this system is also referred to as the Mean Earth/Rotation Axis (MER) system.

The first axis, X_{ME} , is pointing in the mean Earth direction. Y_{ME} is perpendicular to the X and Z axes, forming a right-handed triad. The third axis, Z_{ME} , is parallel to the lunar mean rotational pole. The intersection between the lunar equator and the lunar prime meridian is named the Moon's *mean sub-Earth point*. This concept of "sub-Earth point" originates from the fact that the Moon's rotation is tidally locked to Earth (i.e., the rotation and orbital period are equal). The true sub-Earth point is subject to slight variations in time due to the lunar's orbit eccentricity and other perturbations; thus, a mean point is exploited. In its latest release, this point does not coincide with any crater or surface feature; however, it happens to be close to the Oppolzer A crater. The spherical coordinates for this system, also known as *selenocentric coordinates*, work the very same way of the planetocentric longitude and latitude, ranging from 0° to 360° and from -90° to 90° , respectively.

Principal axes

The Principal Axes (PA) reference system is a lunar body-fixed coordinate system, whose axes are defined by the PA of inertia of the Moon. Their directions are taken from analyses of the Gravity Recovery and Interior Laboratory data. Historically, the PA reference system has been especially useful for dynamical studies on the lunar gravity field and for Lunar Laser Ranging. An ellipsoidal Moon with only a second-degree gravity contribution would have its principal and mean Earth axes parallel. However, third- and higher-degree coefficients of the gravitational model cause a rotation between the PA and ME frames, which leads to a difference of about 1 km on the lunar surface.

The orientation of the ME and PA frames with respect to their inertial counterpart (i.e., LCI) is written as a function of 3 libration angles, whose expressions are simultaneously integrated with the lunar orbital ephemerides.

Three-body synodic and inertial coordinate systems, $X_sY_sZ_s$ and $X_IY_IZ_I$

When three-body problems are analyzed, three-body synodic coordinate systems are commonly used. These reference frames are rotating with the two primary bodies, they are centered at the center of mass of the three-body system (e.g., barycenter of the two primary celestial objects), and the fundamental plane contains the orbits of the primaries. The first axis, X_S , is aligned with the instantaneous rotating vector from the first primary body to the second one. The Z_S axis is in the direction of the angular momentum of the two primary bodies rotating about their common barycenter. The last axis, Y_S , completes the right-handed triad.

An inertial frame, $X_IY_IZ_I$, can also be used in three-body problems. It has the same center and fundamental plane of the synodic reference. The reference directions are aligned with the synodic frame at an epoch. Thus, three-body inertial reference systems are defined fixing the relative position between the primary bodies at a reference time.

Lunar Centered ROTating

The Lunar Centered ROTating (LCROT) frame is a three-body *synodic reference system* centered at the center of the Moon. This reference frame is often used to investigate the behavior of the Cislunar environment in lunar vicinity. This reference frame is constructed translating the origin of the Earth–Moon synodic reference in the Moon’s center. The first axis, X_S ,

is parallel to the instantaneous Earth–Moon direction. The Z_S axis is perpendicular to the Earth–Moon orbital plane, being aligned with the orbital angular momentum of the two primary bodies. The third axis, Y_S , is orthogonal to X_S and Z_S , completing the right-handed triad.

Satellite-based coordinate systems

Many satellite-based reference systems exist, but most of their nomenclature isn't standard, and many systems are developed for specific satellite missions. The fundamental point is that satellite-based coordinate systems are based on the plane of the satellite's orbit. They use the classical orbital elements to describe object locations.

Perifocal coordinate systems, PQW

The perifocal coordinate system, PQW , is a satellite's orbit-based reference frame. It has the fundamental plane on the satellite's orbital plane, and the origin is at the center of the Earth. The P axis points toward perigee, and the Q axis is 90° from the P axis in the direction of satellite motion. The W axis is along the orbital angular momentum. This reference frame is convenient for processing satellite observations.

Satellite coordinate system, RSW (LVLH)

The satellite coordinate system, RSW , moves with the satellite, and it is commonly referred as LVLH (Local Vertical, Local Horizontal). The origin of the LVLH frame is the satellite. The R axis always points out from the satellite along the Earth's radius vector to the satellite as it moves through the orbit (Local Vertical). The W axis is normal to the orbital plane. The S axis completes the right-handed triad. The S axis is perpendicular to the radius vector (Local Horizontal), and it is pointed in the direction of the velocity vector. Note that the S axis is usually not aligned with the velocity vector, except for circular orbits or for elliptical orbits at apogee and perigee.

Concerning the terminology for satellite motion:

- *Radial* motions are parallel to the position vector (i.e., along the R axis).
- *Along-track* or *transverse* displacements are normal to the position vector (i.e., along the S axis).
- *Cross-track* positions are normal to the plane defined by the current position and velocity vectors (i.e., along the W axis).

Some applications define the satellite coordinate frame in different ways. For example, alternative definitions of the LVLH place the primary axis perpendicular to the radius vector, the second axis opposite the angular

momentum vector, and the third axis pointing to the Earth's center. We need to be sure which convention is assumed in the considered application.

The satellite coordinate system can be effectively defined in attitude dynamics as an “airplane-like” reference frame. Indeed, the LVLH can be rotated to define the roll-pitch-yaw axes:

- *Roll* axis is the same as the S axis of RSW (i.e., $+S$).
- *Yaw* axis is opposite to the position vector, points toward the center of the Earth, and also lies in the orbital plane (i.e., $-R$).
- *Pitch* axis is opposite to the angular-momentum vector (i.e., $-W$).

In this book, the RSW is mainly referred according to the above definition and named $\Delta_{ij,k}$. In the comoving frame LVLH-based $\Gamma_{m,n,p}$, differently from $\Delta_{ij,k}$, the radial direction toward the attracting body is called z axis, whereas the y axis is opposite to the angular momentum vector. The following rotation holds from $\Gamma_{m,n,p}$ to $\Delta_{ij,k}$:

$$\Delta \mathbf{R}_\Gamma = \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}.$$

Satellite body coordinate systems, $b_1b_2b_3$

The satellite body coordinate systems, $b_1b_2b_3$, are commonly centered at the center of mass or at the geometrical center of the satellite. Their axes are aligned with notable body-defined directions, such as the geometrical axes or the principal inertia axes. These reference frames are mainly used for attitude dynamics or for GNC applications based on the spacecraft body. There is not a standardization of these coordinate systems, and they are usually developed for specific satellite missions.

The most common satellite body reference frame is the principal inertia one, since it is typically used to describe the spacecraft attitude dynamics. The principal inertia reference frame is centered at the barycenter of the spacecraft, and its axes are aligned with the principal inertia directions, b_1 , b_2 , b_3 , of the rigid body, composing a right-handed triad.

Auxiliary satellite body coordinate systems

Auxiliary satellite body coordinate systems can be defined. They can be centered in relevant points of the spacecrafts (e.g., payload location, center of pressure, sensors, etc.). The axes of these references can be aligned along important reference directions (e.g., solar panels axes, main geometrical

directions, sensor line-of-sight, etc.). They cannot be standardized since they are only defined according to the specific application requirements. However, they should be right-handed triads, and they should be defined by a rigid roto-translation with respect to the main satellite body coordinate system. To avoid confusion, they are introduced in the book immediately after their use, with a unique coordinate naming.



Coordinate transformations

Coordinate transformations can be derived every time the definition of two different reference frames is known with respect to each other's. An alternative to derive the required transformation is knowing the definition of each reference system with respect to a third one.

The transformation matrices exploited to perform elementary rotations between the reference frames use the right-hand rule to define the sign of the rotations. There exist different sequences of rotation angles according to the specific transformation in use. In fact, two reference frames can be aligned exploiting a generic sequence of three rotations around the reference frame axes, and the coordinate transformation is characterized by the specific order of the axes around which the rotations are performed. This concept will be discussed again in the [Chapter 5](#)—Attitude.

Now let's assume a coordinate transformation exploiting three consecutive rotations around the x , y , and z axes. This rotation sequence is denoted as “ x - y - z ,” and it is associated to the following rotation matrices:

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & s_\alpha \\ 0 & -s_\alpha & c_\alpha \end{bmatrix}$$

$$\mathbf{R}_y(\alpha) = \begin{bmatrix} c_\alpha & 0 & -s_\alpha \\ 0 & 1 & 0 \\ s_\alpha & 0 & c_\alpha \end{bmatrix}$$

$$\mathbf{R}_z(\alpha) = \begin{bmatrix} c_\alpha & s_\alpha & 0 \\ -s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $c_\alpha = \cos(\alpha)$, $s_\alpha = \sin(\alpha)$ and α is a generic rotation angle along the relevant axis of the right-handed reference frame. The reader should pay attention to the specific order of the rotation sequence, and, in general, it is advised to work out the elementary rotations for any transformation in use, as it will be done in the next paragraphs. Note that the signs used to compute the rotation sequence depend on the rotation order. If the rotation goes in the positive right-hand rule direction, the angle is positive. On the contrary, the rotation goes in the negative direction and the angle has the opposite sign.

ECI to ECEF

The transformation between the ECI (*GCRF*) and the ECEF (*ITRF*) is performed through a series of rotations that are known as *Earth orientation model*. There are two different approaches to perform such operation: the classical equinox-based transformation and the *Celestial Intermediate Origin* approach. The latter is defined in the IAU2010 convention; a summary of the various contributions is here reported:

- *Polar motion*. It is defined as the motion of the rotation axis with respect to the crust of the Earth. The rotation axis of Earth, perpendicular to the true equator, is named the *Celestial Intermediate Pole* (CIP). Thus, this transformation accounts for the change of basis between the ITRF and the Pseudo-Earth Fixed (PEF) frame.
- *Earth rotation angle or sidereal time*. It considers Earth rotation and allows to switch from the rotating frame to a nonrotating set of axes. The IERS does not provide this angle, but its associated time scale, UT1. In fact, this transformation requires knowledge of the Greenwich apparent sidereal time, which is obtained from the equation of the equinoxes. Formally, the definition allows to switch from the rotating coordinate frame (PEF) to a nonrotating True of Date (TOD) frame.
- *Nutation*. It accounts for the periodic effects primarily due to the Moon, which consist in small oscillations in the Earth's rotation axis. This operation transforms from the TOD to the Mean equator of Date (MOD) frame.
- *Precession*. This transformation accounts for the perturbations due to the Sun, Moon, and other planets' gravitational forces on the Earth's orbit. Specifically, precession results in a slow secular decrement of the ecliptic obliquity (e.g., $0.013^\circ/\text{century}$) and in a westward precession of the equinox (e.g., $0.0033^\circ/\text{century}$). As a result, it causes a roughly circular motion of Earth's rotation axis over each period of precession (26.000 years). This final process converts a vector in the MOD frame

to a vector in the ECI frame. Depending on the approach used, a combined **PN** matrix can be exploited.

The position, velocity, and acceleration transformations take the form of:

$$\mathbf{r}_{ECI} = \mathbf{P}(t)\mathbf{N}(t)\mathbf{R}(t)\mathbf{W}(t)\mathbf{r}_{ECEF} \quad (2.1)$$

$$\mathbf{v}_{ECI} = \mathbf{P}(t)\mathbf{N}(t)\mathbf{R}(t)(\mathbf{W}(t)\mathbf{v}_{ECEF} + \boldsymbol{\omega}_E \times \mathbf{r}_{PEF}) \quad (2.2)$$

$$\mathbf{a}_{ECI} = \mathbf{P}(t)\mathbf{N}(t)\mathbf{R}(t)[\mathbf{W}(t)\mathbf{a}_{ECEF} + \boldsymbol{\omega}_E \times (\boldsymbol{\omega}_E \times \mathbf{r}_{PEF}) + 2\boldsymbol{\omega}_E \times \mathbf{v}_{PEF}] \quad (2.3)$$

with:

$$\mathbf{r}_{PEF} = \mathbf{W}(t)\mathbf{r}_{ECEF} \quad \mathbf{v}_{PEF} = \mathbf{W}(t)\mathbf{v}_{ECEF}$$

where **P** and **N** are the precession-nutation matrices of date t , **R** is the sidereal-rotation matrix of date t , **W** is the polar-motion matrix of date t , and $\boldsymbol{\omega}_E$ represents the rotation rate of the Earth. The complete expressions of these matrices are available in Ref. [6] and in the appendix Chapter 16—Mathematical and Geometrical Rules.

The angles required to perform all these rotations are provided as five Earth Orientation Parameters (EOPs) data, officially released and maintained by the IERS:

- *Universal Time (UT1).* It is a standard time that reflects the average speed of the Earth's rotation, using the prime meridian as a reference point. The excess of the rotation period with respect to the mean period is called *Length of Day*. The difference between UT1 and UTC is indicated as ΔUT1 , and it is periodically corrected to ensure its absolute value always remains below 0.9. This will be discussed with more details in the section about Time.
- *Coordinates of the pole.* The angles that describe the polar motion are expressed as the displacements of the CIP with respect to the *International Reference Pole*, which is the agreed location of the terrestrial pole.
- *Celestial pole offsets.* They provide the difference of the actual celestial motion with the one predicted by the conventional IAU precession/nutation model. Depending on the adopted approach, they are either expressed as the offsets in longitude and in obliquity of the celestial pole ($\Delta\psi$, $\Delta\epsilon$) or as the coordinates of the CIP in the ICRS (dX, dY). The transition between them is reported in the IERS technical note [7].

These values are daily updated by the IERS and published in different bulletins. EOP data are comprehensive of all values since 1962, including 180 days of predictions. Several interpolations are also available.

ECI to PQW

The transformation between ECI and perifocal frame (PQW) is here reported due to its wide use in orbital mechanics. Nevertheless, it is a simple transformation comprising three consecutive rotations of notable angles, which are defined as orbital elements. Indeed, the transformation yields the necessary rotation to express a vector from ECI to a reference frame that is coplanar with the orbit being analyzed.

1. *Right ascension of the ascending node rotation:* it is a rotation along the Z -axis of the ECI reference frame, as shown in Fig. 2.7. The rotation matrix can be expressed as:

$$\mathbf{R}_\Omega = \begin{bmatrix} \cos\Omega & \sin\Omega & 0 \\ -\sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. *Inclination rotation:* it is a rotation along the I' axis of the transformed reference frame, as shown in Fig. 2.8. The rotation aligns the new Z -axis with the orbital angular momentum vector. The rotation matrix can be expressed as:

$$\mathbf{R}_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & \sin i \\ 0 & -\sin i & \cos i \end{bmatrix}$$

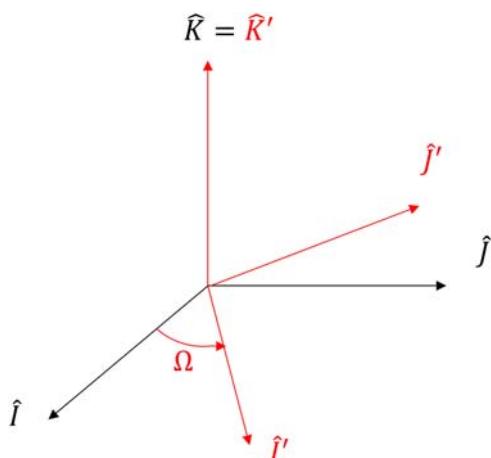


Figure 2.7 RAAN rotation from ECI to PQW.

3. *Argument of pericenter rotation:* it is a rotation along the I' axis of the transformed reference frame, as shown in Fig. 2.9. The rotation aligns the new X-axis with the periapsis. The rotation matrix can be expressed as:

$$\mathbf{R}_\omega = \begin{bmatrix} \cos \omega & \sin \omega & 0 \\ -\sin \omega & \cos \omega & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Hence, the final transformation from ECI to PQW or vice versa can be expressed as:

$${}_{PQW}\mathbf{R}_{ECI} = \mathbf{R}_\omega \mathbf{R}_i \mathbf{R}_\Omega$$

$${}_{ECI}\mathbf{R}_{PQW} = {}_{PQW}\mathbf{R}_{ECI}^T = \mathbf{R}_\Omega^T \mathbf{R}_i^T \mathbf{R}_\omega^T$$

ECI to RSW (LVLH)

Similar to the previous paragraphs, an important and common reference frame transformation is the one used to express vectorial quantity from ECI to RSW (or LVLH) frame. We recall that the RSW is a comoving frame attached to the spacecraft center of mass. Let us assume that the orbital position and velocity in the ECI frame is given by the vector $\mathbf{r}(t)$ and $\mathbf{v}(t)$, and the constant specific angular momentum is calculated as

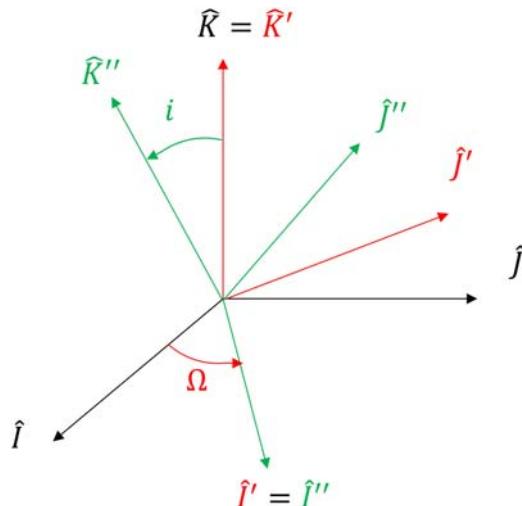


Figure 2.8 Inclination rotation from ECI to PQW.

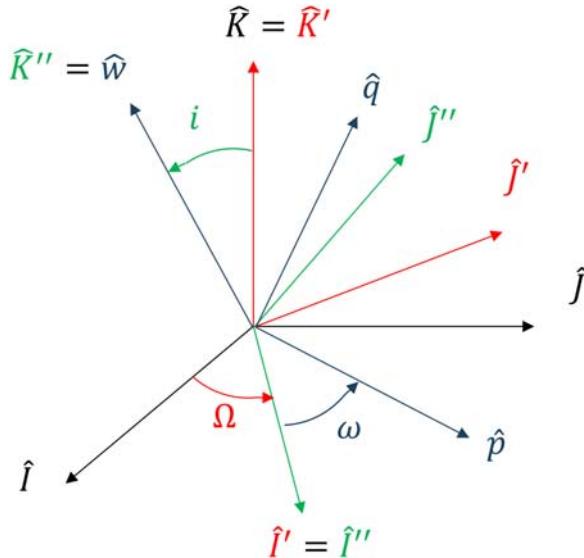


Figure 2.9 Argument of perigee rotation from ECI to PQW.

$\mathbf{h} = \mathbf{r}(t) \times \mathbf{v}(t)$. Using the definition of the RSW reference frame, we can express the rotation matrix between ECI to RSW, and vice versa, as:

$${}_{RSW}\mathbf{R}_{ECI} = \begin{bmatrix} \bar{\mathbf{r}} \\ \bar{\mathbf{h}} \times \bar{\mathbf{r}} \\ \bar{\mathbf{h}} \end{bmatrix}$$

$${}_{RSW}\mathbf{R}_{ECI} = {}_{RSW}\mathbf{R}_{ECI}^T$$

where $\bar{\mathbf{r}}$ and $\bar{\mathbf{h}}$ are the normalized unit vectors.

Being the RSW reference frame defined based on the orbital state of the spacecraft, it is possible to derive the rotation matrix from RSW to ECI using the orbital elements of the spacecraft orbit, as:

$${}_{RSW}\mathbf{R}_{ECI} = \begin{bmatrix} \cos(\nu)\cos(\Omega) - \sin(\nu)\cos(i)\sin(\Omega) & -\sin(\nu)\cos(\Omega) - \cos(\nu)\cos(i)\sin(\Omega) & \sin(i)\sin(\Omega) \\ \cos(\nu)\sin(\Omega) + \sin(\nu)\cos(i)\cos(\Omega) & -\sin(\nu)\sin(\Omega) + \cos(\nu)\cos(i)\cos(\Omega) & -\sin(i)\cos(\Omega) \\ \sin(\nu)\sin(i) & \cos(\nu)\sin(i) & \cos(i) \end{bmatrix}$$

where $\nu = \omega + \theta$ is the true latitude, with θ the true anomaly.



Time

The main purpose of time is to define with precision the moment of an event. This moment is referred to as the epoch of the event; thus, the epoch designates a particular instant described as a date. The date is defined as a certain time interval elapsed from a reference epoch. For instance, the days passed from a reference event, or the years, months, days, hours, minutes, and seconds from the year zero. To have a practical time system, we need a precise, repeatable time interval measurement, which is based on some physical phenomenon that we can easily measure. Current time-keeping for scientific, engineering, and general-purpose applications are based on:

- Sidereal time.
- Solar time.
- Dynamical time.
- Atomic time.

Sidereal time and solar time are based on the Earth's rotation and are related through mathematical relationships. Dynamical and atomic time are independent from the other forms.

Solar time is loosely defined by successive transits of the Sun over a local meridian, while sidereal time is defined as the time between successive transits of the stars over a specific meridian. However, the apparent motion of celestial objects in the sky is not regular, and so we had to define a constant pace time reference. The concept of universal time, UT, was adopted years ago. It's based on a fictitious mean Sun with uniform motion in right ascension along the equator. This fictitious mean Sun is now defined mathematically as a function of the sidereal time. So, ultimately, we derive UT from sidereal time.

Dynamical time measures time by analyzing the motion of celestial bodies, such as the Earth's motion about the Sun. In this case, the measure of the time is associated to the ephemerides of the celestial objects. Terrestrial time, TT, and TDB are common dynamical time references, which are equivalent for the vast majorities of applications.

Finally, whenever accurate time measurements are needed, atomic time shall be used. The International Atomic Time, TAI, is based on counting the cycles of a high-frequency electrical circuit maintained in resonance with a cesium-133 atomic transition. One SI (International System) second equals the duration of 9 192 631 770 periods of the wavelength associated with the radiation emitted by the electron transition between two hyperfine levels of

the ground state of cesium-133 at 0° K. TAI achieves a precision that permits the observation of relativistic effects for clocks in motion or accelerated by a local gravitational field.

Universal time

Going back to the most common UT, we shall consider that there are three distinct realizations of UT. For precise applications, we must distinguish between UT, UT0 and UT1, even if the differences are small. Note that Greenwich mean time, (GMT), the local time in England, is not equivalent to UT. For this purpose, it's useful to introduce procedures that determine the various forms of UT and to distinguish their differences.

At first, we must define the UT as the mean solar time at Greenwich. The mean solar time is derived from measurements of the Earth orientation with respect to the apparent Sun. Then, the difference between apparent solar time and mean solar time can be computed from the equation of time, which describes the discrepancy between these two kinds of solar time [8]. Note that the apparent solar time is the interval between successive transits that we observe from a particular longitude, and thus the length of each apparent day differs by a small amount.

UT0 is found by reducing the observations of stars from many ground stations. This is done because the motion of the Sun (i.e., UT) cannot be measured with enough precision, while the apparent motion of the stars, or radio galaxies can be tracked with great accuracy. UT0 is calculated as 12 h plus the Greenwich Hour Angle. Indeed, when the Greenwich hour angle is 0° (i.e., the Sun is exactly over the prime meridian), the UT0 shall be 12:00.

Then, we correct UT0 for polar motion, so the time is independent of station location to obtain UT1. Differences from UT0 are typically about 30 milliseconds, and they can be computed exploiting the Earth orientation model and the tabulated EOP. However, UT1 is commonly tabulated and given to be directly used in GNC applications. Moreover, UT1 is the reference time for reference frame conversion calculation.

The most commonly used time system is Coordinated Universal Time, *UTC*, which is derived from an ensemble of atomic clocks. It's designed to follow *UT1* within ±0.9s:

$$\Delta UT1 = UT1 - UTC.$$

Because *UT1* varies irregularly due to variations in the Earth's rotation, we must periodically insert leap seconds into *UTC* to keep the two different

time scales in close agreement. *UTC* is the basis of civil time systems and is on ordinary clocks. The local time in England, *GMT*, is thus associated to the time zone *UTC+00:00*, and not the solar definition of the *UT*.

Satellite often uses GPS time. This time system began with the introduction of the GPS system operational capability in 1980: the GPS epoch is January 6, 1980, 00:00 UTC. GPS time is not adjusted, so it differs from *UTC* by the number of leap seconds:

$$GPS = UTC + \Delta AT - 19.0s.$$

ΔAT represents the number of leap seconds posing the difference between *GPS* and *UTC* times, and 19s is the delta between *GPS* time and atomic clock time, *TAI*:

$$GPS = TAI - 19.0s.$$

As a result, atomic time relates to *UTC* as:

$$TAI = UTC + \Delta AT.$$

UTC always differs by an integer number of leap seconds from *TAI* (i.e., ΔAT), but the two reference times are maintained on a daily basis to keep them within one microsecond from each other, although they are usually much closer.

The *TT* also relates to *TAI* by a constant offset as:

$$TT = TAI + 32.184s.$$

Looking at the offsets between the reference times, graphically represented in Fig. 2.10, we can note that the difference in atomic time, ΔAT , remains constant until changed with a leap second, whereas the difference in coordinated and universal time, $\Delta UT1$, changes continuously. Time offsets to the GNSS/GPS satellites are needed when determining navigation information.

Julian dates

Time intervals are managed in *UTC* counting years, months, days, hours, minutes, and seconds from a reference epoch, the beginning of Christian era. This results in a date format composed by six variables, with different time interval meanings. Moreover, the calendar dates contain periodic cycles with discrete steps through the addition of leap years and seconds. To avoid these problems, time for spacecraft applications is conveniently managed exploiting the JDs. The JD is a continuous interval of time measured in days from the epoch January 1, 4713 B.C., 12:00. JDs are precisely

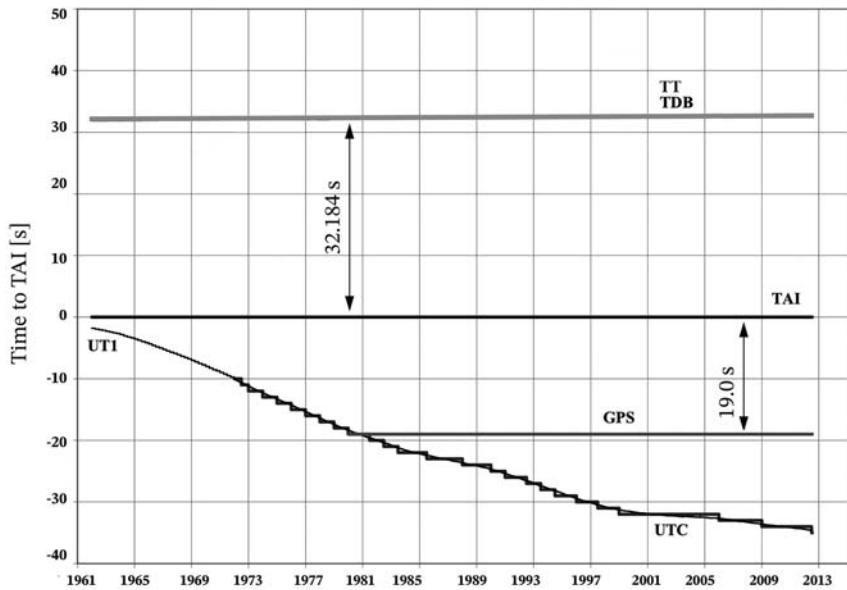


Figure 2.10 Reference time conversion with respect to TAI.

365.25 days per year (i.e., a leap year every four years). Note that the convention starts each JD at noon each day.

The JD values are typically very large, so we may be tempted to use only a few decimal places. However, because the units are days, it is suggested to retain at least eight decimal digits to provide reasonable accuracy (i.e., about 4×10^{-4} s). The most precise approach for GNC and numerical applications is to split the JD into a day and fractional day part.

Finally, the IAU recommends using a Modified Julian Date, *MJD*, commonly calculated as follows:

$$MJD = JD - 2400000.5.$$

MJD reduces the size of the date by about two significant digits, and it can reduce potential confusion because it begins each day at midnight instead of noon. Moreover, additional MJDs can be defined for specific applications. For instance, JDs since GPS epoch (i.e., GPS-JD), or since specific recent epochs (i.e., J2000-JD, J2020-JD) can further improve the reduction of the significant digits of the date.

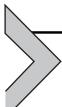
Few commonly used epochs are:

GPS epoch = 2 444 244.5 = January 6, 1980–00:00:00.000 UTC

J2000.0 = 2 451 545.0 = January 1, 2000–12:00:00.000 TDB (TT)

$$J1900.0 = 2\ 415\ 021.0 = \text{January 1, 1900} - 12:00:00.000 \text{ TDB (TT)}$$

As desired, the JD provides a continuous, simple, concise method of preserving year-month-day-hour-minute-second information in one variable, which is especially nice for computer and GNC applications. However, the time accuracy reduction due to JD large numbers shall be considered, and proper numerical techniques shall be implemented to avoid too coarse time values.



What is relevant for GNC?

Planetary models, reference systems, and time intervals are among the fundamental elements on which every GNC subsystem is based. Hence, they shall be taken in great consideration since the beginning of the GNC design.

In these regards, the accuracy of the models, reference frame conversions, and timekeeping shall be in agreement with the accuracy of the whole GNC system. It is pointless to convert reference frames with accuracies down to the millimeter, and then have navigation precision in the order of the kilometer. Similarly, maintaining the deviation with respect to UTC within one microsecond is not wise if the on-board clock accuracy is one second. These considerations are valid for any section of the GNC design, but they are fundamental when dealing with standard references and basic models.

The used references shall be selected in agreement with each other, and in accordance with the prescription of the international standards. The specific references to be used to model the environment will be discussed in [Chapter 3](#)—The Space Environment. However, to make some examples, the IGRF geomagnetic model shall be used together with the WGS-84 ellipsoid, similarly the EGM gravitational models shall be used with the WGS-84 geoid, while EIGEN-GL04 gravity model shall be used in agreement with its geoid definition. The European designs are invited to use the latter, while in the United States, the former is used. Also, the time references shall be selected in agreement with the used models. For instance, the astronomical ephemerides typically use the number of Julian centuries from J2000, while the reference frame rotations use UT1 and the TT. In these regards, the relations between the reference times shall be considered, and the time conversions shall be performed with the accuracy required by the GNC system.

Time accuracy is one of the critical aspects to consider when dealing with GNC and with reference frame conversion. In fact, considering that a

spacecraft is orbiting with a velocity in the order of km/s, errors in the order of milliseconds immediately result in few meters of additional errors. This has even a deeper impact if a reference frame conversion is performed inside the GNC functions. In fact, in this case, the errors can be magnified because the position, plus the time-induced errors, is rotated in a reference frame which is not coincident with the correct one because of the time delay of the conversion. Thus, accurate GNC system shall take into account the time offsets $\Delta UT1$ and ΔAT . The latter is seldom modified, but the former is frequently updated, and dedicated uploads of the new values shall be planned.

This last point opens to the general comment on the on-board availability of ancillary data used to set-up reference models and conversions. In fact, EOPs, IGRF, and geopotential coefficients, as well as updated reference time offsets values shall be known to guarantee the proper calculations in the GNC functions. Moreover, these values shall be known with an accuracy level which is in agreement with the accuracy of the GNC. To make some examples, neglecting $\Delta UT1$ correction leads to error in the order of 100 m in position and 0.02 m/s in velocity. Even worse if GPS time is directly associated to UTC or UT1, since the resulting error is in the order of 10 km and 10 m/s. A simple ECI-ECEF rotation, without EOP data, introduces an error in the order of 30 km and 40 m/s. Considering the precession reduces the error to 200 m and 0.25 m/s, while adding the nutation effects brings down the error to 15 m and 0.1 m/s. If continuous updates of EOP are not possible (i.e., an update frequency of one month would be suggested), linear interpolation for these parameters can be used [9]. In this case, using interpolated EOP, the reference frame transformation errors are in the order of 2 m and 0.002 m/s. Note that a good interpolation is possible for polar motion, precession, nutation, and length of the day, while $\Delta UT1$ cannot be effectively interpolated. Then, $\Delta UT1$ should be updated with a frequency in the order of one month.

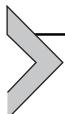
A final consideration shall be dedicated to numerical precision in managing these quantities. In fact, most of the GNC software is based on single-precision floating-point values, since double-precision floating-point numbers are more demanding in terms of computational resources. However, reference frame and time conversion should be performed in double precision. In fact, despite single precision is intrinsically responsible for a position accuracy not better than 0.5 m in Launch and Early Orbit phase, due to float truncation errors, single precision ECI-ECEF transformation may lead to 10 km and 50 m/s spike errors in the converted data.

Hence, reference frame conversion should be computed in double precision, and then the output converted to single precision, if this downscaling is needed. The numerical precision is an issue also for timekeeping in JDs. If the JD is not divided in integer and fractional part, the single precision values can have a maximum resolution not better than few hours in the present days. Using MJDs with respect to more recent epochs (e.g., GPS-JD or J2020-JD) can improve the single precision date resolution to a few seconds. The best practice is to divide any JD in integer and fractional parts. In this case, considering the single precision data type, according to IEEE Standard 754, the day units can have a resolution of $\sim 1 \times 10^{-7}$ d, corresponding to around 0.01 s. Anyway, it is evident how double-precision floating-point numbers should be used for JDs operations, and in general for all timekeeping functions. Furthermore, it is globally suggested not to use JDs in GNC applications, but to use MJD, GPS-JD, or the JD at the epoch of the IGRF model (i.e., IGRF-13 to be used with J2020-JD).

References

- [1] ICAO, World Geodetic System—1984 (WGS-84) Manual, 2002.
- [2] F.G. Lemoine, S.C. Kenyon, J.K. Factor, et al., The Development of the Joint NASA GSFC and NIMA Geopotential Model EGM96, NASA Goddard Space Flight Center, Greenbelt, Maryland, 20771 USA, July 1998.
- [3] K.P. Nikolaos, A.H. Simon, C.K. Steve, K.F. John, EGM2008: the development and evaluation of the earth gravitational model 2008 (EGM2008), *Journal of Geophysical Research: Solid Earth* 117 (B4) (April 2012), <https://doi.org/10.1029/2011JB008916>.
- [4] ECSS-E-ST-10-04C, Space Engineering – Space Environment, 2008.
- [5] E. Gregersen, Cavendish Experiment, Encyclopedia Britannica, 17 January 2019. Accessed 15 March 2022, <https://www.britannica.com/science/Cavendish-experiment>.
- [6] D.A. Vallado. Fundamentals of Astrodynamics and Applications, Space Technology Library, Microcosm Press, 2013.
- [7] G. Petit, B. Luzum, IERS Conventions (2010), IERS Technical Note No. 36, in: IERS Technical Note No. 36, International Earth Rotation and Reference Systems Service, Frankfurt am Main, 2010.
- [8] D.W. Hughes, B.D. Yallop, C.Y. Hohenkerk, The equation of time, *Monthly Notices of the Royal Astronomical Society* 238 (June 15, 1989) 1529–1535.
- [9] D.A. Vallado, T.S. Kelso, Earth orientation parameter and space weather data for flight operations, in: 23rd AAS/AIAA Space Flight Mechanics Meeting, American Institute of Aeronautics and Astronautics (AIAA), 2013. AAS 13-373.

This page intentionally left blank



The space environment

**Andrea Capannolo¹, Emanuele Paolini², Andrea Colagrossi¹,
Vincenzo Pesce³, Stefano Silvestrini¹**

¹Politecnico di Milano, Milan, Italy

²D-Orbit, Fino Mornasco, Italy

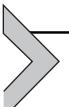
³Airbus D&S Advanced Studies, Toulouse, France

All the bodies in space interact with the surrounding environment: the space environment. This strongly influences the design of a space mission, for what concern the materials to use, the suitable electronical components, and, in general, most of the design choices leading to an operative spacecraft. Regarding the guidance, navigation, and control (GNC) system, the most relevant aspects of the space environment to consider are the perturbation forces and torques. Whenever a spacecraft is on orbit, it is subject to the influence of many perturbation sources that affect its dynamics. Hence, the GNC system shall manage these effects in order to properly control the spacecraft's motion. This chapter explains the concept of perturbation, discussing the major external and internal perturbations sources. External perturbations are those properly related with the surrounding space environment, while internal perturbations are related with the elements inside the spacecraft, which anyway influences the satellite dynamics. The combination of the two imposes the complete space environment. As a matter of fact, the internal perturbation sources are characterized by the fact the satellite is in space and not on ground. This chapter is composed by the following sections:

- *Perturbation sources.* This section has the purpose of introducing the main perturbation sources influencing the spacecraft dynamics.
- *External perturbations.* In this section, the external perturbations effects are introduced. Gravitational and magnetic contributions, atmospheric drag, solar radiation pressure (SRP), and third-body perturbations are described.
- *External perturbations modeling guidelines.* The main guidelines to model the presented external perturbations, while designing a GNC system, are given in this section.
- *Internal perturbations.* The internal perturbative sources are detailed in this section. Flexibility and sloshing are introduced as well as electromagnetic

disturbances, internal vibrations, thermal snap, and parasitic forces and torques due to thruster firing and plume impingement.

- *Internal perturbations modeling guidelines.* The main guidelines to model the presented internal perturbations, while designing a GNC system, are given in this section.
- *What is relevant for GNC?* In this section, the influence of external and internal perturbations on the GNC chain is carefully explained.



Perturbation sources

The equations of Keplerian orbital motion, which will be discussed in [Chapter 4](#) – Orbital Dynamics, rely on the strong assumption that an orbiting body is only affected by the attractor's gravity, and that such gravity is a perfect central force field. In the real case, this is not true, since the orbital environment is filled with various force fields, not always of gravitational nature. Such forces (or accelerations) are, in most cases, of several orders of magnitude smaller than the gravity from the main attractor, and for this reason, they are labeled as “perturbations”; however, their modeling is paramount if accurate simulations and GNC design are desired.

In general, the perturbative accelerations may be taken care of through a simple summation in the external forces term of the second law of motion. In particular, the equation of orbital perturbed dynamics reads ([Chapter 4](#) – Orbital Dynamics):

$$\ddot{\mathbf{r}} = -\frac{Gm}{r^3} \mathbf{r} + \mathbf{a}_p$$

where \mathbf{a}_p is the sum of all the considered perturbative forces. It is relevant to note that any perturbation source interacts with the spacecraft that is an extended body with distributed mass, and, thus, it does not only produce a resulting force but also a resulting torque. Thus, both the translational and rotational motion are affected by the perturbations. This chapter describes the main features of the perturbation sources, and their specific effects on the orbital and attitude dynamics will be discussed in the related chapters ([Chapter 4](#) – Orbital Dynamics, [Chapter 5](#) – Attitude Dynamics).

The GNC designer shall know how the perturbations interact with the spacecraft in order to implement a system that is capable to counteract the forces and torques influencing the dynamics. Moreover, it shall be capable to understand which perturbation sources are the most relevant to be considered and modeled. The goal of this chapter is to learn the fundamental concepts leading to the solution of these problems.



External perturbations

External perturbations are those effectively associated with the space environment surrounding the spacecraft, and they are associated to physical phenomena already present in space. The most relevant external perturbation sources in space applications are:

- Gravity irregularities from nonspherical attractors ($\mathbf{a}_{\text{gravity}}$).
- Magnetic fields (\mathbf{a}_{mag}).
- Drag from interaction with atmospheres (\mathbf{a}_{drag}).
- Pressure of incoming solar radiation (\mathbf{a}_{sdp}).
- Gravity from other celestial objects (\mathbf{a}_{3bp}).

Although a full inclusion of all perturbation sources always provides the most accurate results, it is often sufficient to consider some perturbations only, as the others may have a negligible effect on a specific orbit. Indeed, their orders of magnitude greatly change with the environment type (i.e., mass of the attractor, presence of an atmosphere, presence of a liquid metal core, distance from other celestial bodies, in-light/in-shadow orbital motion, etc.), and with the distance of the orbiting body from the main attractor. Fig. 3.1 shows an example of the orders of magnitude for Earth orbits. The GNC design shall take into account only those perturbations are actually relevant, and the understanding of the relative magnitudes of the perturbation terms is crucial. For example, from Fig. 3.1, it is evident that a GNC

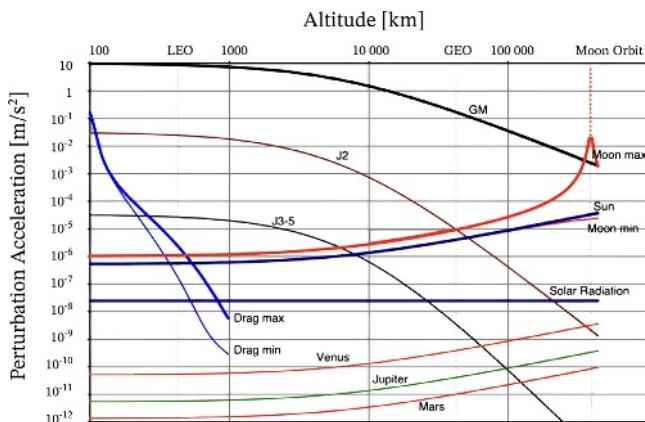


Figure 3.1 Order of magnitude of perturbations as function of the distance from Earth center of mass. Courtesy: K. Yazdi, E. Messerschmid, *Analysis of parking orbits and transfer trajectories for mission design of cis-lunar space stations*, Acta Astronautica 55 (3–9) (2004) 759–771.

design in low Earth orbit (LEO) shall account more for atmospheric drag and the first harmonics of the gravity field, while in geostationary Earth orbit (GEO), it shall consider the gravity attraction of the Moon and Sun more than the other terms. It can be remarked here that considering the atmospheric drag contribution above a 1000 kilometers is pointless.

In the next sections, each perturbation source is explained, and its formulation derived. The discussion is valid and can be generalized to the space environment about any celestial body, but it is focused on perturbation models for Earth applications.

Gravity field of a central body

The most simple approximation, when dealing with gravitational attraction, is to consider the celestial objects as perfect spheres. This enormously simplifies the construction of dynamical models, as the gravitational force can be easily expressed through an elementary analytical formulation.

Let's consider the general formulation of the *gravitational potential*:

$$U = G \iiint_m \frac{1}{r} dm \quad (3.1)$$

where G is the gravitational constant, dm an infinitesimal mass portion of the attracting body, and r_p the distance between the control point (or spacecraft) and such infinitesimal mass. According to the *shell theorem* [1], the potential of a perfect sphere is equivalent to the one of a dimensionless point, where all the attractor's mass has been shrunk in the center of the sphere, hence Eq. (3.1) becomes:

$$U = \frac{Gm}{r},$$

which is the point mass gravitational potential, and its gradient leads to the point mass (or spherical) gravitational acceleration vector:

$$\nabla U = -\frac{Gm}{r^3} \mathbf{r}$$

In practice, any celestial object possesses some irregularities and oblateness, which acts as a perturbation that adds up to the point mass gravity. To describe such irregularities in the field, let's consider again Eq. (3.1). Now, the integral has to be solved for a finite distribution of mass.

This can be done through the *Spherical Harmonics Expansion (SHE)* model, by approximating the integral quantity $\frac{1}{r_p}$ as a series of *Legendre's Polynomials* [2]:

$$\frac{1}{r_p} \sim \sum_{k=0}^{\infty} \frac{r_m^k}{r^{k+1}} P_k(\cos(\beta)) \quad (3.2)$$

where r_m and r are the positions of the infinitesimal mass and of the control point, respectively, from the attractor's center of mass (CoM); β is the angle between the two positions vectors, and P_k is the k th-degree Legendre's polynomial, expressed in the *Rodrigues' formula* form as:

$$P_k(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n$$

By substituting (3.2) into (3.1), and by developing the integral, the new gravitational potential for an irregular gravity field is obtained [3]:

$$U = \frac{Gm}{r} \left\{ 1 + \sum_{i=2}^n \sum_{j=0}^i \left(\frac{R}{r} \right)^i [C_{ij} \cos(j\lambda) + S_{ij} \sin(j\lambda)] P_{ij}(\cos(\theta)) \right\} \quad (3.3)$$

Here, m and R are the mean mass and radius of the attractor, θ and λ represent the colatitude and longitude of the point where the potential is being evaluated, C_{ij} and S_{ij} are the *normalized Stokes coefficients*, and $P_{ij}(\cdot)$ are the *associated Legendre's polynomials*, which read:

$$P_{ij}(x) = (1 - x^2)^{\frac{i}{2}} \frac{d^j}{dx^j} P_j(x)$$

The Stokes coefficients are instead expressed in terms of the infinitesimal mass element and its related quantities (radius, colatitude, and associated polynomial of the longitude):

$$\begin{aligned} C_{ij} &= \frac{1}{m} \iiint \left(\frac{r_m}{R} \right)^i \frac{(i-j)!}{(i+j)!} P_{ij}(\sin(\theta_m)) \cos(j\lambda_m) dm S_{ij} \\ &= \frac{1}{m} \int \int \int_m^m \left(\frac{r_m}{R} \right)^i \frac{(i-j)!}{(i+j)!} P_{ij}(\sin(\theta_m)) \sin(j\lambda_m) dm \end{aligned} \quad (3.4)$$

Typically, a scaling factor N_{ij} is introduced [4] to reduce machine overflow and underflow during computation:

$$N_{ij} = \sqrt{(2 - \delta_{ij})(2i + 1) \frac{(i-j)!}{(i+j)!}} \quad (3.5)$$

Using (3.5) to scale the coefficients from Eq. (3.4), the *normalized Stokes coefficients* are obtained:

$$\begin{aligned}\overline{C}_{ij} &= \frac{1}{m} \sqrt{\frac{(2 - \delta_{0j})(i-j)!}{(2i+1)(i+j)!}} \iint_m \left(\frac{r}{R}\right)^i P_{ij}(\sin(\theta)) \cos(j\lambda) dm \overline{S}_{ij} \\ &= \frac{1}{m} \sqrt{\frac{2(i-j)!}{(2i+1)(i+j)!}} \iint_m \left(\frac{r}{R}\right)^i P_{ij}(\sin(\theta)) \sin(j\lambda) dm\end{aligned}$$

To avoid altering the result of Eq. (3.3), the scaling factor is also applied to the associated Legendre function $\left(\overline{P}_{ij} = \frac{P_{ij}}{N_j}\right)$.

From Eq. (3.3), one can notice that the first term is the point mass gravitational potential. Since we are here interested in the perturbative forces only, the point mass term can be omitted. Also, from the second equation of (3.4), it is observed that when $j = 0$, then $S_{ij} = 0$. Given these considerations, the gravity potential expression can be reorganized as follows:

$$U_{\text{gravity}} = \frac{Gm}{r} \left\{ \sum_{i=2}^n \overline{C}_{i0} \left(\frac{R}{r}\right)^i \overline{P}_i(\cos(\theta)) + \sum_{i=2}^n \times \sum_{j=1}^i \left(\frac{R}{r}\right)^i [\overline{C}_{ij} \cos(j\lambda) + \overline{S}_{ij} \sin(j\lambda)] \overline{P}_{ij}(\cos(\theta)) \right\} \quad (3.6)$$

The C_{i0} terms are independent from the longitude and are related to the colatitude only. They are commonly referred to as *zonal harmonics*, and typically contain the most relevant terms of the irregular mass distribution of the attractor. An alternative expression of this (nonnormalized) coefficients is J_i and relates to the previous one as follows:

$$J_i = -C_{i0}$$

Notice that in the case of purely axisymmetric objects, zonal harmonics are the only source of irregularity in the gravitational field. The other terms describe the *Sectorial Harmonics* (when $i = j$), dependent on longitude only, and the *Tesseral Harmonics* (when $i \neq j \neq 0$), which are affected by both latitude and longitude.

To obtain the acceleration generated by the irregular field's potential, its gradient is taken. In particular, we are interested in defining the acceleration in cartesian coordinates, despite the potential of Eq. (3.6) is function of spherical coordinates' quantities (radius, longitude, colatitude). Hence,

defining $\mathbf{r} = [x, y, z]$ as the vector of cartesian coordinates of the field point, the acceleration can be expressed as:

$$\mathbf{a}_{\text{gravity}} = \nabla U = \frac{\partial U}{\partial r} \left(\frac{\partial r}{\partial \mathbf{r}} \right)^T + \frac{\partial U}{\partial \theta} \left(\frac{\partial \theta}{\partial \mathbf{r}} \right)^T + \frac{\partial U}{\partial \lambda} \left(\frac{\partial \lambda}{\partial \mathbf{r}} \right)^T$$

By developing the partial derivatives, and omitting the point mass gravity, the perturbative acceleration becomes:

$$\mathbf{a}_{\text{gravity}} = \begin{bmatrix} \left(\frac{1}{r} \frac{\partial U}{\partial r} - \frac{z}{r^2 \sqrt{x^2 + y^2}} \frac{\partial U}{\partial \theta} \right) x - \left(\frac{1}{x^2 + y^2} \frac{\partial U}{\partial \lambda} \right) y \\ \left(\frac{1}{r} \frac{\partial U}{\partial r} - \frac{z}{r^2 \sqrt{x^2 + y^2}} \frac{\partial U}{\partial \theta} \right) y - \left(\frac{1}{x^2 + y^2} \frac{\partial U}{\partial \lambda} \right) x \\ \left(\frac{1}{r} \frac{\partial U}{\partial r} - \frac{z}{r^2 \sqrt{x^2 + y^2}} \frac{\partial U}{\partial \theta} \right) x - \left(\frac{1}{x^2 + y^2} \frac{\partial U}{\partial \lambda} \right) y \end{bmatrix}$$

The potential of Eq. (3.6) assumes that the attractor has a fixed shape and mass distribution. If a very high precision in modeling perturbations is required, it is desirable to include the effect of *tides*, for both the solid and the liquid masses characterizing the attractor. To account for tides, the Stokes coefficients must be varied in time, according to the motion of the main objects causing the tide (e.g., in the case of Earth, the major contributions are given by the gravity of Moon and Sun). In the specific case of Earth, it is useful to distinguish between *solid Earth tides* and *ocean tides*.

Solid Earth tides consider the coefficients variation due to the deformation of the solid part of the planet. After labeling the Moon and the Sun with the indexed $q = 2$ and $q = 3$, the variation of the coefficients is modeled as follows:

$$\begin{aligned} \Delta C_{ij} &= \frac{k_{ij}}{2i+1} \sum_{q=2}^3 \frac{m_q}{m} \left(\frac{R}{R_q} \right)^{i+1} P_{ij}(\cos(\theta)) \cos(j\lambda) \Delta S_{ij} \\ &= \frac{k_{ij}}{2i+1} \sum_{q=2}^3 \frac{m_q}{m} \left(\frac{R}{R_q} \right)^{i+1} P_{ij}(\cos(\theta)) \sin(j\lambda) \end{aligned}$$

where k_{ij} are the *Love numbers* [5]. Such expression is valid for all j when $i \in [2, 3]$ and for $J \in [0, 2]$ when $i = 4$. Details about the model can be found in Ref. [6], where additional minor effects such as *pole tide* (tide from Earth's rotation) are also described.

The ocean tides cause a smaller variation in mass distribution than solid Earth tides. Their effect on the coefficients is described by the following equations:

$$\begin{aligned}\Delta C_{ij} &= \frac{4\pi R^2 \rho_w}{m} \left(\frac{1 + k'_{ij}}{2i + 1} \right) \sum_{q=1}^2 \frac{m_q}{m} \left(\frac{R}{R_q} \right)^{i+1} P_{ij}(\cos(\theta)) \cos(j\lambda) \Delta S_{ij} \\ &= \frac{k'_{ij}}{2i + 1} \sum_{q=1}^2 \frac{m_q}{m} \left(\frac{R}{R_q} \right)^{i+1} P_{ij}(\cos(\theta)) \sin(j\lambda)\end{aligned}$$

where k'_{ij} are the *load deformation coefficients*, and ρ_w is the density of seawater. Again, details of the ocean tides can be found in Ref. [6]. Note that in LEO, solid Earth tides are the same order of magnitude of Sun gravity attraction, and they are more relevant than SRP. At GEO altitude, they can be neglected in the vast majority of the applications. Ocean tides are always less relevant than solid Earth tides, and they typically account for about 10%–15% of their magnitude.

Gravitational models

Gravity field modeling is dependent from the selection of the coefficients to insert into the spherical harmonic expansions. In fact, there are different gravitational models with different accuracy levels. The most relevant are:

- Joint Gravity Model (JGM-3)—Order: 70×70 —1996.
- Earth Gravity Model 96 (EGM-96)—Order: 360×360 —1997.
- Earth Gravity Model 08 (EGM-08)—Order: 2190×2190 —2008.
- European Improved Gravity model of the Earth by New techniques 04 (EIGEN-GL04C)—Order: 360×360 —2006.

The first three are maintained by the US entities, while the latter is developed by an effort of European contributions, and it is standard for GNC applications in Europe [7].

Their order of expansion is quite above the typical needs for GNC applications. Indeed, GNC accuracy requirements drive the maximum degree and order. For instance, 4×4 fields are often adequate for deep-space orbits, while some low Earth satellites need around 50×50 field. However, the most accurate and updated models are suggested, since they also improve the accuracy of low-order coefficients. In these regards, EGM-08 or EIGEN-GL04C is preferred.

Note that, as remarked in Chapter 2 — Reference Systems and Planetary Models, we must consistently use a gravitational model with its associated

constants. Only in this way, it is possible to obtain documented accuracy, since mixing constants from different theories and models leads to errors and inconsistencies.

For reader's convenience, the first three zonal harmonics of the Earth's gravity field (EGM-08) are reported here:

- $J_2 = 0.0010826261\overline{7385}$.
- $J_3 = -0.0000025324$.
- $J_4 = -0.0000016198$.

Note how J_2 is by far the strongest perturbation due to the Earth's shape. It is almost 1000 times larger than the next largest coefficient (J_3).

Magnetic field

The magnetic field acts as a perturbation through the interaction with charged particles on the orbiting object. It is worth mentioning that this kind of perturbation has a minor effect on the translatory motion, while has a more significant role in the rotational dynamics.

The magnetic field \mathbf{B} can be expressed as the gradient of a potential function U_m :

$$\mathbf{B} = \nabla U_{mag}$$

The magnetic potential can be expressed in terms of a spherical harmonic expansion, which resembles the gravitational potential formulation, and it reads:

$$U_{mag} = R_0 \sum_{i=1}^n \left(\frac{R_0}{r}\right)^{i+1} \sum_{j=0}^i [\bar{g}_{ij} \cos(j\lambda) + \bar{h}_{ij} \sin(j\lambda)] \bar{P}_{ij}(\cos(\theta))$$

Here, \bar{g}_{ij} and \bar{h}_{ij} are the *Schmidt coefficients* which are tabulated data referred to the *International Geomagnetic Reference Field* (IGRF) model [8]. The bar over the Schmidt coefficients indicates a normalization, which however is different from the one performed in the gravitational case, and results in:

$$\bar{g}_{ij} = \Pi_{ij} g_{ij}$$

$$\bar{h}_{ij} = \Pi_{ij} h_{ij}$$

$$\Pi_{ij} = \sqrt{\frac{(2 - \delta_{0j})(i - j)!}{(i + j)!}} \frac{(2i - 1)!}{(i - j)!}$$

For what concern magnetic field modeling, the IGRF is the standard model accepted worldwide. The IGRF describes the large-scale structure of the Earth's main magnetic field and its secular variation. It covers a significant time span, and so it is useful for interpreting historical data and to perform forecasts within its validity time interval. The IGRF is updated at five-year intervals, reflecting the most accurate measurements and predictions available at the release time. For example, the 13th edition of the IGRF model (IGRF-13 or IGRF-2020) was released in December 2019, and it is valid from 1900 until 2025. The IGRF assumes $R_0 = 6\,371.2$ km, being the geomagnetic conventional Earth's mean reference spherical radius, but it shall be handled (e.g., to convert between geocentric and geodetic coordinates) over the WGS-84 ellipsoid with major equatorial radius equal to 6378.137 km. The order of truncation of the harmonic expansion is $n = 13$. The IGRF has been produced and updated under the direction of the International Association of Geomagnetism and Aeronomy (IAGA) since 1965.

Atmospheric drag

In case of relatively low-altitude orbits around bodies with an atmosphere, even the slightest, rarefied presence of gaseous particles causes a long-term deviation in the trajectory of an orbiting object. Such effect is still orders of magnitude below the nominal gravitational attraction of the celestial body; hence, it can be considered as a perturbative force.

The formula of the atmospheric drag, acting on the orbiting body, follows the classical expression from fluid dynamics [9], and reads:

$$\mathbf{a}_{drag} = -\frac{1}{2} \frac{c_D A}{m} \rho v_{rel}^2 \hat{\mathbf{v}}_{rel} \quad (3.7)$$

Note that Eq. (3.7) can be applied to the entire spacecraft, in a simplified modeling approach, or it can be subdivided in more elemental terms composing the body, to be summed at the end to find a more accurate perturbing force on the spacecraft. The latter finite element method-like approach guarantees improved results, at the cost of increased computational burden.

The *drag coefficient* c_D scales the drag effect according to the properties of the orbiting body (e.g., its shape). For practical space engineering application, a common preliminary approach is to set such value to 2.2, which derives from a flat plate model.

A and m are, respectively, the section area of the object facing the *relative wind* and its mass. Their ratio indicates how a low-mass and extended body is subjected to drag perturbative acceleration significantly more than a small, massive body. The drag coefficient, mass, and section area appearing in Eq. (3.7) are often collected and describe as a single quantity, namely the *ballistic coefficient*:

$$BC = \frac{m}{c_D A}.$$

From previous considerations, one can understand the sensitivity of the orbiting body to drag force by evaluating this single quantity: a large ballistic coefficient implies a low effect of drag perturbation on the body, while a low value implies a higher contribution of such acceleration to the spacecraft dynamics.

The *relative wind velocity* v_{rel} , and its unit vector $\hat{\mathbf{v}}_{rel}$, shall not be confused with the orbital velocity. In fact, the atmosphere has its own motion, mainly given by the planet's rotation (and, for a minor part, from winds). Hence, the relative wind speed can be expressed as:

$$\mathbf{v}_{rel} = v_{rel} \hat{\mathbf{v}}_{rel} = \frac{d\mathbf{r}}{dt} - \boldsymbol{\omega} \times \mathbf{r}$$

with \mathbf{r} being the orbiting body's position in a rotating frame jointed to the attractor's rotation, and $\boldsymbol{\omega}$ being the attractor's angular velocity vector.

The final, and perhaps most complex, term of Eq. (3.7) is the *atmospheric density* ρ . The actual density depends on several factors, and it is a quantity varying with time, geographical location, and altitude. Because of the complex behavior, several models have been developed to provide an acceptable and increasingly accurate representation of the atmosphere. Fig. 3.2 displays the development flow of the currently available models, sorted by subgroup (according to the model's method and assumptions) and the year of development.

Most models rely on the *ideal gas law* and the *hydrostatic equations*, respectively:

$$\rho = \frac{pM}{gRT}$$

$$\Delta p = -\rho g \Delta h$$

where p is the local pressure, M the mean molecular mass, g the gravity acceleration, R the universal gas constant, T the local temperature, ρ the

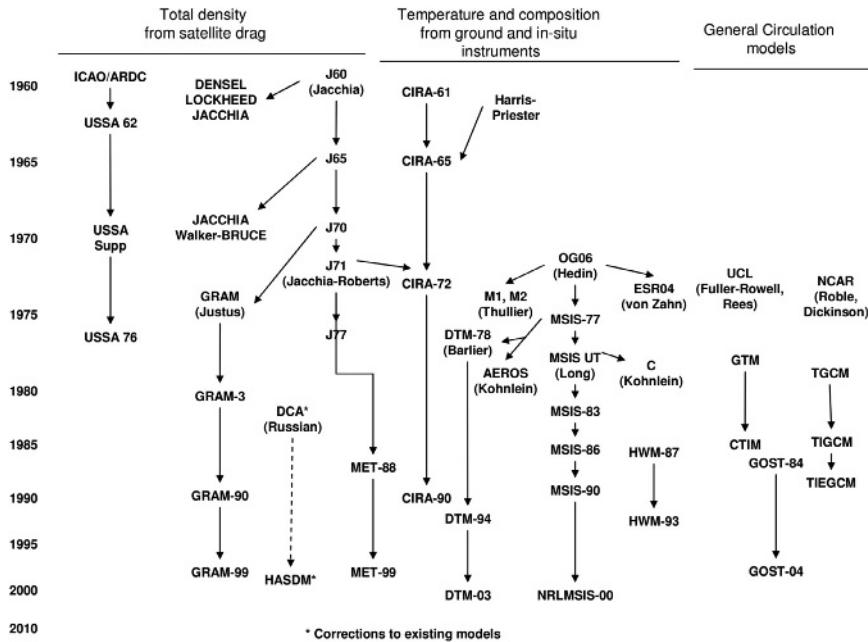


Figure 3.2 Development scheme of available atmospheric models. Courtesy: D.A. Val-lado, D. Finkelman, *A critical assessment of satellite drag and atmospheric density modeling*, *Acta Astronautica* 95 (2014) 141–165.

local density, and h the altitude. Nevertheless, a major distinction is made between *static models* and *time-varying models*. Static models provide relatively simple formulas and are the easiest to implement. Despite the static nature, some of them implement variations as function of the local coordinates (latitude and longitude). Typically more accurate, the time-varying models implement atmospheric variations from several nonstatic factors, such as solar cycles and fluxes, day–night cycles, etc.

Relevant examples of relatively simple atmospheric models are the *exponential model*, the *US Standard Atmosphere* and *CIRA models* [10,11], and the *Jacchia models* [12,13]. A detailed listing and description of available models can be found in Ref. [14].

The *exponential model* represents the simplest approach to describe the atmosphere's quantities and relies on the following analytical expression:

$$\rho = \rho_0 e^{-\frac{h-h_0}{H}} \quad (3.8)$$

with ρ_0 and h_0 being the reference density and altitude, respectively, and H the *scale height* (fractional change in density with height). Despite its

simplicity, the model is not suitable for a proper evaluation of the drag, as Eq. (3.8) poorly describes the true density variation with height.

A more accurate representation of density (suitable for preliminary/general studies) is provided by the *US Standard Atmosphere* and the *CIRA models*. Such models leverage the same formula from (3.8) but implement different reference values (ρ_0 , h_0 , H) for different portions of the atmosphere.

The Jacchia models, instead, leverage empirical temperature profiles and analytical expressions to retrieve the atmospheric parameters as function of position, time, solar, and geomagnetic activity, thus providing more accurate results than the previous models. The Jacchia models are also recommended by the international standards, with specifications on the limitations and the parameters to properly set-up the model. In particular, Jacchia-Roberts (J71 and successive) and Jacchia-Bowman (JB-2006) are the most used atmospheric models for LEO applications (i.e., 150–1000 km).

Accurate atmospheric models require input to correctly model solar and geomagnetic activity. Indeed, direct collisions of the solar wind and air particles interacting with the Earth's geomagnetic field heat the atmosphere varying the density–altitude profile. The commonly used geomagnetic planetary index, K_p , is a quasilogarithmic, worldwide average of geomagnetic activity below the auroral zones. The geomagnetic planetary amplitude, a_p , is a linear equivalent of the K_p index, and it can be averaged on a daily basis. For what concern solar activity, the contribution of solar flux to atmospheric density is mainly from incoming solar radiation. Solar flux can be related to incoming solar radiation analyzing the wavelength of 10.7 cm. Regular measurements of F10.7 exist from about 1940. Daily values are regularly distributed worldwide, as well as 81-day average values (i.e., three solar rotations).

These geomagnetic and solar parameters need to be known to perform correct atmospheric analyses. Note that their values are usually correlated, since the daily planetary amplitude tends to follow the 11-year cycle of sun-spots, although consistently large maxima of a_p usually occur in the declining phase of each 11-year cycle of F10.7. For this reasons, geomagnetic storms, if not properly taken into account, represent a threat for spacecraft operations. In February 2022, a strong geomagnetic storm caused the increase of drag at the low altitudes that prevented the 40 SpaceX Starlink satellites from begin orbit-raising maneuvers and led to their premature reentry [15].

Solar radiation pressure

The SRP typically represents one of the most relevant perturbation sources when far from the attractor and in deep space. The resulting acceleration

exerted on the spacecraft (or any orbiting object) depends on several factors, which can be collected into two subgroups, i.e., Sun properties and spacecraft properties.

Sun properties are related to time variations of solar activity and solar cycles and involve the spectrum of the emitted radiation as well as its intensity. In practice, such properties are averaged through the *solar flux* (SF [W/m^2]), a constant value which varies with the distance from Sun only, except for minor variations depending on the period within the solar cycle. A legacy value for Earth orbits is 1367 W/m^2 (still used for many applications), although various measurements across the years provided a range of solar fluxes spanning from below 1360 W/m^2 to above 1370 W/m^2 . More accurate models may rely on F10.7 measurements and predictions.

The corresponding pressure exerted by the radiation is directly obtained by dividing the flux per the speed of light c , namely:

$$p_{srp} = \frac{SF}{c}$$

which, in case of Earth missions, translates to $4.57 \times 10^{-6} \text{ N/m}^2$.

Concerning the spacecraft, several factors have to be considered to properly compute the perturbing acceleration, namely the extension and orientation of spacecraft's sides facing the Sun, and their (time-variant) absorption and reflection properties. In practice, there are different levels of depth in the SRP perturbation model.

The simplest is the so-called *cannonball model*. As the name suggests, it considers the spacecraft as a sphere, thus removing dependencies from the attitude of the spacecraft itself. In this way, a single section A (the largest section of the sphere) can be considered. Furthermore, the model does not distinguish between specular and diffusive reflection and considers only the transparency, reflectivity, and absorptivity of the object. The cannonball model equation reads:

$$\mathbf{a}_{srp} = -\frac{p_{SRP} C_R A}{m} \hat{\mathbf{r}}_{Sun} \quad (3.9)$$

with $\hat{\mathbf{r}}_{Sun}$ being the Sun direction from the spacecraft, m the spacecraft mass, and C_R the *reflectivity coefficient*. From Eq. (3.9), one can understand that low-mass and wide objects are subjected to a larger perturbation from the solar radiation. Also, only a radial perturbation from the Sun can be computed with this model.

The reflectivity coefficient C_R spans from 0 to 2 and defines how the transparency/reflectivity/absorptivity properties of the object affect the momentum exchange between the radiation and the object itself. A value of zero indicates a completely transparent object, which does not exchange momentum. A value of 1 represents a fully absorbing object (a black body), so the full radiation momentum is transferred to the object. A value of 2, instead, represents a fully reflecting object, which doubles the incoming momentum from the radiation.

Despite the easy implementation of the cannonball model, one may need a more detailed representation of the SRP perturbation, by considering both the actual shape of the spacecraft and its specular and diffusive reflection properties. To do so, consider each surface of the spacecraft, with its own orientation (normal vector $\hat{\mathbf{n}}$) and absorptivity/reflectivity properties. It is possible, for each surface, to separately compute the accelerations due to the absorption (\mathbf{a}_{SRP}^a), diffused reflection (\mathbf{a}_{SRP}^d), and specular reflection (\mathbf{a}_{SRP}^s) of the radiation:

$$\begin{aligned}\mathbf{a}_{srp}^a &= -\frac{p_{srp}c_a A}{m} \cos(\phi) \hat{\mathbf{r}}_{Sun} \\ \mathbf{a}_{srp}^d &= -\frac{p_{srp}c_d A}{m} \cos(\phi) \left(\frac{2}{3} \hat{\mathbf{n}} + \hat{\mathbf{r}}_{Sun} \right) \\ \mathbf{a}_{srp}^s &= -2 \frac{p_{srp}c_s A}{m} \cos^2(\phi) \hat{\mathbf{n}}\end{aligned}\quad (3.10)$$

where ϕ is the angle between the Sun direction and the surface normal, and c_a, c_d, c_s are the *absorptivity*, *diffusive reflectivity*, and *specular reflectivity coefficients*, such that:

$$c_a + c_d + c_s = 1 \quad (3.11)$$

The overall acceleration from SRP is then the summation of the three components from (3.10), for every lit surface “ i ” of the spacecraft. By collecting the components aligned with $\hat{\mathbf{n}}$ and $\hat{\mathbf{r}}_{Sun}$, and using (3.11) to eliminate the absorptivity coefficient, the total acceleration reads:

$$\mathbf{a}_{srp} = - \sum_{i=1}^N \frac{p_{SRP} A_i}{m} \cos(\phi_i) \left[2 \left(\frac{c_{ri}}{3} + c_{si} \cos(\phi_i) \right) \hat{\mathbf{n}} + (1 - c_{si}) \hat{\mathbf{r}}_{Sun} \right] \quad (3.12)$$

where N is the total number of lit surfaces of the spacecraft. Notice that a further level of depth can be obtained by using (3.12) with variable coefficients with respect to other quantities (such as surface temperature), when such relations are available.

Eclipse

The SRP force is directly dependent on the solar flux SF . If eclipse occurs, the value of the Sun luminosity has to be modified accordingly. A simple way to implement eclipses is by considering a conical model. Let's consider the following vectors as in Fig. 3.3:

- $\mathbf{r}_B = -\mathbf{r}_{sc}|_P$ vector from vehicle to planet.
- $\mathbf{r}_S = -\mathbf{r}_{sc}|_P + \mathbf{r}_{Sun}|_P$ vector from vehicle to Sun.

Then, the following quantities can be computed:

- $H_B = \arcsin\left(\frac{R_p}{|\mathbf{r}_B|}\right)$ apparent planet radius as seen by the vehicle.
- $H_S = \arcsin\left(\frac{R_{Sun}}{|\mathbf{r}_S|}\right)$ apparent Sun radius as seen by the vehicle.
- $\gamma = \arccos\left(\frac{\mathbf{r}'_B \cdot \mathbf{r}_S}{|\mathbf{r}_B||\mathbf{r}_S|}\right)$ angular separation between planet and Sun as seen by the vehicle.

R_p and R_{Sun} are the radii of considered body and Sun, respectively.

Given H_B , H_S , and γ , the necessary and sufficient eclipse conditions (illustrated in Fig. 3.4) are:

- No eclipse:

$$\gamma > H_B + H_S$$

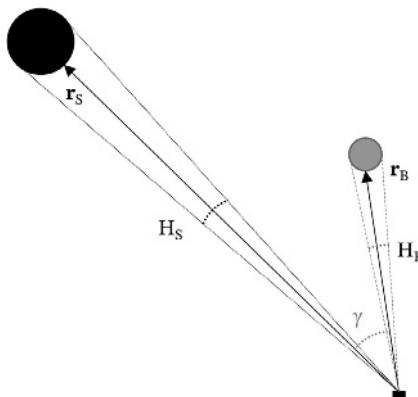


Figure 3.3 Eclipse geometry.

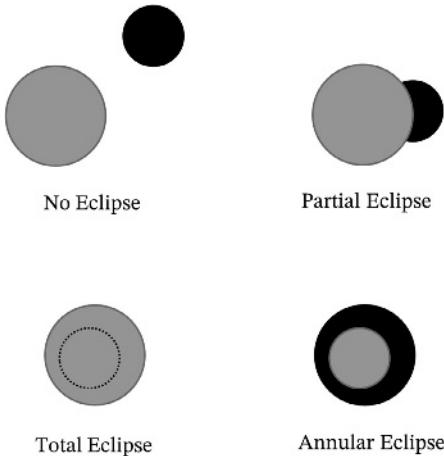


Figure 3.4 Eclipse conditions.

- Total eclipse:

$$H_B > \gamma + H_S$$

- Annular eclipse:

$$H_S > \gamma + H_B$$

- Partial eclipse: if there is an eclipse not belonging to any of the previous categories.

The Sun luminosity changes for the four cases, and it can be expressed as ratio with respect to the ideal case I_0 . The luminosity ratio during eclipse conditions is computed as:

- No eclipse: $\frac{I}{I_0} = 1$
- Total eclipse: $\frac{I}{I_0} = 0$
- Annular eclipse: $\frac{I}{I_0} = 1 - \frac{1-\cos(H_B)}{1-\cos(H_S)}$
- Partial eclipse:

$$\frac{I}{I_0} = 1 - \frac{\left(\pi - \cos H_S \cos \left(\frac{\cos H_B - \cos H_S \cos \gamma}{\sin H_S \sin \gamma} \right) - \cos \phi_h \cos \left(\frac{\cos H_S - \cos H_B \cos \gamma}{\sin H_B \sin \gamma} \right) - \cos \left(\frac{\cos \gamma - \cos H_S \cos H_B}{\sin H_S \sin H_B} \right) \right)}{\pi - (1 - \cos H_S)}$$

Albedo and infrared emission

Solar radiation immediately reflecting off the Earth—or off any planet—back to space is called albedo. The Earth reflects almost 30% of the incoming

solar radiation. The remaining solar radiation is absorbed and reemitted at a later time as infrared (IR) radiation or emissivity. The emitted IR energy at the Earth's surface (IR) is about 237 W/m^2 . Both terms can be measurable on some satellites. Albedo and emitted IR are commonly associated to specific wavelengths, each of which transfer through the atmosphere differently.

Albedo and IR-induced accelerations on the spacecraft are computed as for the direct SRP, with Eq. (3.12). However, first we have to compute ground albedo and emissivity, and also consider each region of the Earth that is visible to the satellite, or to one satellite flat plate surface. Obviously, albedo and emitted IR are directed and oriented from the Earth surface and not from the Sun. The albedo and emissivity at the ground can be modeled as spherical harmonic expansions with respect to the geoid [16] or as simple expressions only function of the altitude.

Third-body perturbation

When distance from the main attractor is sufficiently large, a relevant source of disturbance is provided by the additional gravitational pulls from other massive bodies (i.e., third-body, fourth-body, etc.). To model these contributions to satellite's acceleration, one may start by recalling the equation of the N-body problem (Chapter 4 – Orbital Dynamics), where this time, all the relevant masses are preserved within the formulation.

By shifting the origin to the main attractor's CoM, the following equation is obtained:

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3} \mathbf{r} + \sum_{i=1}^n \mu_i \left(\frac{\mathbf{r} - \mathbf{r}_i}{||\mathbf{r} - \mathbf{r}_i||^3} - \frac{\mathbf{r}_i}{r_i^3} \right) = -\frac{\mu}{r^3} \mathbf{r} + \mathbf{a}_{3bp} \quad (3.13)$$

where μ_i represents the individual constant of the i-th additional gravity source, and \mathbf{r}_i its position vector with respect to the main attractor.

Notice that the sum in the right-hand side of Eq. (3.13) contains the gravity pull of the additional source to the spacecraft, plus the gravitational pull of the main body to the additional source. This formulation may cause numerical problem in case of a third (fourth, fifth, etc.) body which is at very large distance from both the main attractor and the spacecraft, like the Sun for an Earth-orbiting object, such that $||\mathbf{r} - \mathbf{r}_i|| \sim r_i$. In this situation, the second term of Eq. (3.13) will have two very small terms (the two ratios), which are in turn subtracted from one another, making this term prone to numerical error.

Although this issue is not always relevant and should be evaluated case by case, there are some workarounds that have been developed to solve the problem. A direct analytical solution to the problem has been provided, and it can be found in Ref. [17]. Other techniques leverage series expansions. A straightforward approach in this sense is to leverage a Taylor series, as shown in Ref. [18]. For example, if the series is stopped at the second order, the following expression holds:

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3} \mathbf{r} + \frac{\mu_3}{r_3^3} \left[\mathbf{r} - 3\mathbf{r}_3 \frac{\mathbf{r} \cdot \mathbf{r}_3}{r_3^2} - \frac{15}{2} \left(\frac{\mathbf{r} \cdot \mathbf{r}_3}{r_3^2} \right)^2 \mathbf{r}_3 \right]$$

which is obtained by assuming $r \ll r_3$. Despite the numerical stability, this formulation may still lead to relevant errors due to the approximated formulation itself. Although still approximated, a more accurate approach is described in Ref. [19], where the norm $\|\mathbf{r} - \mathbf{r}_3\|$ is expressed through the cosine law, enabling the usage of Legendre polynomials as done for irregular gravity fields (Eq.3.2). Details on the Legendre-based expansion of third-body perturbation can also be found in Ref. [14].

Ephemerides

When modeling third-body perturbations, it is fundamental to have a right representation for the relative position of the involved celestial objects. This is done by exploiting planetary ephemerides, which are time-dependent expressions of the deterministic evolution of the celestial objects' positions. The ephemerides of the Jet Propulsion Laboratory are considered as the most accurate, but we can use less precise formulations from the Astronomical Almanac [20] or Meeus [21].

In general, planetary ephemerides use interpolating polynomials in terms of the classical orbital elements and the elapsed Julian centuries to provide the planet's position and velocity vector at a given epoch in a given frame. They are created by numerically integrating the planetary dynamics accounting for the mutual influence of the planets. The numerical integration commonly uses a variable step-size that averages about 7 h. Once the ephemerides are created, the results are fit with Chebyshev polynomials and collected into few days blocks (e.g., 4, 8, 16, or 32 days). The general accuracy of the ephemerides is about $0.01''$. The Moon ephemerides are accurate to about 2 m ($0.001''$), and the Sun is accurate to about 200 m ($0.0003''$).

Data are available for the planets (e.g., Development Ephemeris, DE-405—1998, DE-430—2013) and the Moon (e.g., Lunar Ephemeris, LE-

405—1998, LE-430—2013). The reported versions are among the ones suggested by the international standards, even if more recent ephemerides data have been released. For the sake of completeness, the classical derivation of Chebyshev polynomials for ephemeris approximation is described here.

Chebyshev polynomials

Chebyshev polynomials of the first kind are usually adopted. The general definition of the Chebyshev polynomial $T_n(x)$ of the first kind in x (interpolation points) of degree n is defined by the relation [22]:

$$T_n(x) = \cos(n\vartheta) \text{ when } x = \cos(\vartheta).$$

From this general definition, it is possible to derive the fundamental recurrence relation of the Chebyshev polynomials:

$$T_0(x) = 1, \quad T_1(x) = x$$

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x), \quad \forall n \geq 2.$$

This general expression is valid in the range $[-1, 1]$ of x . In order to generalize the expression for an interval $[a, b]$, the following variable has to be introduced:

$$s = \frac{2x - (a + b)}{b - a}$$

with a and b being the limits (or boundaries) of the interval. The Chebyshev polynomials of the first kind appropriate to $[a, b]$ are thus $T_n(s)$.

Coefficients computation

In order to compute the interpolating Chebyshev coefficients, the following steps are implemented:

- The reference position γ of the body is computed from reference ephemeris files (e.g., JPL, Almanacs).
- Interval limits a and b , and interpolation points x are defined.
- The normalized Chebyshev polynomials $T_n(s)$ up to the desired order n are computed.
- All the polynomials are collected in a vector p , and the following equation is solved:

$$c = p^T \gamma.$$

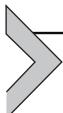
Once the coefficients are computed, the Chebyshev interpolation can be performed at each time step to retrieve the approximated body position.

Chebyshev interpolation

The final Chebyshev interpolation is performed according to the classical formula:

$$f(t) = \sum_{i=0}^n c_i T_i(s_t).$$

where $f(t)$ is the approximated body position at time t .



External perturbations modeling guidelines

To help the GNC designer, modeling the space environment, international standards, and guidelines have been developed. Several standards exist and are variously applied worldwide. For what concern European standards, the perturbation modeling guidelines are collected by the “European Cooperation for Space Standardization” (ECSS) in a dedicated document [7].

The main aspects and standards for the described perturbations are reported in the following paragraphs.

Gravity

The gravitational potential, described in Eq. (3.6), allows to compute the gravitational attraction of a nonspherical object, for any point in space and with any desired accuracy, by choosing the coefficients and the truncation level of the expansion. Nevertheless, it is desirable to minimize the computational effort for evaluating the potential, by properly selecting the useful coefficients for the specific application (distance from the attractor and location of the orbit).

The ECSS standards provide a guideline for this selection. Two main aspects have to be considered when dealing with truncation of the series:

- The “attenuation factor” $\left(\frac{R_0}{r}\right)^i$ from Eq. (3.6) rapidly decreases as the distance from the attractor increases, leading to a lesser influence of the local irregularities of the attractor itself (the uneven gravity model approaches the point mass model).
- Contributions that have a lower order of magnitude with respect to the inherent model noise level can safely be neglected.

From these observations, a rule-of-thumb was developed by Kaula [4] to obtain the order of magnitude of normalized expansion coefficients as function of the degree i :

$$\bar{C}_{ij}, \bar{S}_{ij} = \frac{10^{-5}}{i^2} \quad (3.14)$$

By multiplying the Kaula term in Eq. (3.14) with the attenuation factor for every degree i , one gets the remaining signal power of the corresponding harmonic term. Once such value becomes sufficiently lower (i.e., one order of magnitude less) than the inherent noise level of the model, the corresponding degree can be selected as a truncation level for the expansion. For example, at the eighth order expansion, the Kaula term is 1.5×10^{-7} , and an orbit with radius equal to 25,000 000 km has an attenuation factor at the eighth order of 1.5×10^{-5} . The remaining signal power results 2.25×10^{-12} . Then, for a 360×360 model, the inherent noise level can be approximated by Eq. (3.14) as 7.7×10^{-11} . Since the order of magnitude of the remaining signal power is lower than the noise level of the model, an eighth order expansion would be sufficient for the considered orbit. In practice, being the Kaula's rule an approximation, adding few more degrees is strongly suggested, having a minor impact on the computational burden. For instance, a 12×12 expansion would be ideal for the example case.

A second aspect that should be considered is the presence of resonances between the attractor's rotation and the orbital motion. In this situation, certain harmonic components are continuously sensed by the satellite in exactly the same way, and even very small harmonic components that are in exact phase with orbital motion may then result in significant orbital perturbations after sufficient propagation time intervals. If such resonances exist (i.e., as in repeated orbits, where the ground track of the spacecraft returns to the same point on the Earth surface after M orbital revolutions), some terms of the expansion, even if negligible according to Kaula's rule, may lead to the long-term perturbation. Therefore, such isolated terms shall be identified and included in the expansion.

Magnetic field

Magnetic field shall be modeled according to the available international geomagnetic models. The last version of IGRF shall be used as the internal geomagnetic field model (i.e., main geomagnetic field due to Earth internal phenomena). The IGRF shall use the standard parameters for ellipsoid and

reference radius definition, and geodetic coordinates calculations, as described in the previous sections. For times in the past and in the future, IGRF shall be used in association with its secular variations. However, the validity time interval shall not be exceeded.

Internal geomagnetic field is the primary (i.e., >90%) magnetic field under quiet solar and geomagnetic activity conditions. However, currents flowing in the ionosphere induce an external magnetic field component. Regarding the modeling of the average Earth external magnetic field, and its variations due to geomagnetic and solar activity, the international guidelines provide two standard models to choose from:

- Alexeev's model [23].
- Tsyganenko and Stern [24].

Atmospheric models

The critical aspect of modeling the atmospheric drag perturbation is the choice of the model for the attractor's atmosphere, which determines temperature and pressure profiles with altitude. Despite the large number of available models, the international guidelines indicate two specific models to be used for Earth:

- The NRLMSISE-00 [25] model *shall* be used for calculating the neutral temperature, the total density, and the detailed composition of the atmosphere.
- The JB-2006 [26] model or JB-2008 [27] model *may* be used for calculating the total atmospheric density above an altitude of 120 km.

The NRLMSISE-00 model shall not be mixed with the JB-2006 model.

Furthermore, the guidelines provide information on how to deal with cyclic atmospheric properties due to interactions with solar radiation and geomagnetic activity. In the document, a table containing inputs for the aforementioned models is provided to cover “long-term” and “short-term” effects.

Also, additional information is provided for Earth wind models, as well as for other planets' atmospheres. For the details, the reader is invited to read Ref. [7].

Solar radiation

The international guidelines do not provide information about the models for SRP; however, they indicate which value should be considered for the total solar irradiance.

In particular, despite the wide use of a value around 1367 W/m^2 , the document indicates an average value at one astronomical unit of distance from the Sun of 1361 W/m^2 , with one-sigma uncertainty of $\pm 0.5 \text{ W/m}^2$. In addition to that, a long-term smoothed solar cycle is considered, leading to a minimum value of 1360.5 W/m^2 and to a maximum value of 1362 W/m^2 . In addition to that, the international guidelines report the oscillation due to the distance of the Earth from the Sun. At the aphelion, the total solar irradiance is 1316 W/m^2 , while at the perihelion 1407 W/m^2 .

For what concern Earth's albedo and IR-emitted radiation, international guidelines suggest to consider their variability across the globe, function of the cloud coverage, and surface properties. Average albedo and IR values shall be used with care only for short duration analyses.

Third-body perturbation

The two main aspects to select, when including the third-body perturbation, are which attractors to consider, and what model to rely on for their trajectories.

The international guidelines provide some suggestion in the specific case of Earth-orbiting satellites. In particular, it is stated that the only significant attraction (i.e., other than the Sun and the Moon) arises from Venus, Mars, Jupiter, and Saturn, as the other bodies are either too small or too far away. In this case, all the attractors, other than the Earth, can be modeled as point masses without visible losses in accuracy.

Regarding the bodies' trajectories, the standards indicate that Development Ephemerides data on planets (DE-430) and Lunar Ephemerides data (LE-430), both given in Ref. [28], shall be used.



Internal perturbations

In addition to external disturbances, caused by environmental elements, a spacecraft is subject to several sources of internal disturbances for the dynamics, which distinguish from the previous as being caused by elements that are part of the satellite. As such, spacecraft designers have the possibility to manipulate them through design choices, leading to requirements constraints, in order to limit the influence of these perturbations. The most common internal disturbances are:

- Flexibility.
- Sloshing.
- Parasitic forces and torques during thrusters firing.
- Electromagnetic disturbances.
- Internal vibrations.
- Parasitic forces and torques due to plume impingement.
- Thermal snap effects.

Internal perturbations modeling is very difficult to be generalized, since it is very mission-specific and related to characteristics of the spacecraft. To understand the effects and the magnitude of the internal perturbation sources, preliminary models shall be applied on the spacecraft dynamics in order to have a general overview of the spacecraft internal environment. Then, we shall select the relevant internal perturbations in order to implement the refined models for the design analyses. The following sections give an overview on how the internal perturbations sources interact with the spacecraft.

Flexibility

Deep studies have been developed about the control of rigid bodies since the very beginning of the space era, but far inferior efforts have been concentrated on the control of rigid bodies with flexible appendages until the mid-90s, when the problem became of large interest. In general terms, the necessity to take into account spacecraft flexibility depends on the level of accuracy required to the GNC subsystem, since every spacecraft is flexible at some extent. The growing interest in the field is due to several factors: first, the increasing performance of payloads reflects in a higher power demand, thus in larger solar panels, up to a point where flexibility of such panels has a nonnegligible influence on the satellite dynamics. Second, increasing performances also mean more stringent requirements on GNC in terms of control error (i.e., pointing error) and stability. Moreover, missions are increasingly requiring fast maneuvering to acquire different targets in limited time, and therefore the satellite experiences higher accelerations that can excite flexibility.

Flexible appendages (e.g., solar panels, large antennas, etc.) modify the dynamics of the satellite from the simple rigid-body case: indeed, the flexible appendages interact with the main body of the spacecraft, creating a different overall response to external disturbances or to commanded forces and torques. The movement of the propellant inside the tank (i.e., sloshing) causes the same effects, but it will be treated separately in the next paragraph. The presence of vibrations can perturb both the orbital motion and the attitude

one. However, in terms of orders of magnitude, the variation of position of the satellite's CoM in the inertial frame due to vibrations is usually negligible in comparison with the tolerance on the orbit altitude imposed, for example, by the ground track. In fact, the energy of the orbital motion is extremely large, and the attitude and flexible dynamics have negligible effect upon the orbital one. Moreover, the attitude-control frequencies are closer (i.e., more influenced) to those of the flexible structures, and this may further increase the coupling between attitude and flexible dynamics. Thus, the following discussion will focus on the attitude perturbations due to flexibility, yielding to a simplification decoupling between flexible and orbital motions. This is generally true for most internal perturbations. Flexible effects, and other internal disturbs, can be extended to the CoM dynamics for very advanced and peculiar studies.

The difference in complexity between flexible dynamics and rigid dynamics is evident: the flexible problem is not only infinite dimensional for the structural flexibility but also highly nonlinear. Often, the problem can be simplified considering a discrete parameters model: in this way, the infinite number of dimensions involved is reduced to that of the primary modes of vibrations taken into account [29]. The following approaches can be considered for the aim of modeling spacecraft flexibility [30]:

- *Distributed parameters.* The solar array, or extended appendage, is represented by a cantilever beam, whose deformations are expressed in continuous coordinates. The Euler–Bernoulli equation is used, considering an appendage deflection, δ , as the product of a time-dependent function (i.e., generalized coordinate) and a space-dependent term (i.e., mode shape):

$$\delta(l, t) = \sum_{i=1}^n \phi_i(l) p_i(t)$$

being n the number of flexible modes taken into account. Partial derivatives equations are obtained for the deflection of the beam, ordinary derivatives equations for the spacecraft dynamics [31,32]. A Ritz–Galerkin approximation is often used to represent the solar array or the flexible structure, as it will described in the following.

- *Discrete parameters.* Every appendage is a punctual mass interconnected by a spring. In this case, ordinary differential equations are obtained both for the flexible part and the spacecraft. Each mass–spring system represents a mode of vibration for the appendage [33]. A similar approach can be used

for modeling the sloshing effects. The damping factor of the spring can only be measured on a spacecraft prototype; for simulations, a value in the range of 0.01%–0.1% of the critical damping is typically used as a conservative approach.

- *N-bodies modeling.* The structure is seen as N interconnected rigid bodies, each of which is a mass with its own inertia. Multibody dynamics modeling approaches and theories are used to represent and analyze the whole flexible system [34].
- *Finite elements methods (FEMs).* This approach combines the two first techniques together. The finite elements may be modeled separately as spring-mass elements or as “distributed parameters.” The FEM techniques are then applied to model and simulate the flexible elements on the spacecraft [29].

In any case, flexible effects are not accounted in the dynamics as additional perturbation terms, but the aforementioned approaches are used to derive flexible equations of motion, which shall be used in place of the rigid body ones. Hence, the spacecraft dynamics including the flexibility effects shall be modeled with flexible dynamical equations plus the terms due to the other external and internal perturbation sources [35].

Example of a discrete parameters modeling

In the following, the approach presented in Ref. [33] is reported as an example, where the equations of motions of a spacecraft with reaction wheels and flexible solar panels are derived from the Lagrange equations in terms of quasicoordinates:

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \omega} \right) + [\boldsymbol{\omega}_\times] \left(\frac{\partial L}{\partial \omega} \right) = \mathbf{t},$$

where \mathbf{t} is the torque vector, L the Lagrangian function, and $[\boldsymbol{\omega}_\times]$ the cross-product matrix associated to the angular rate of the spacecraft:

$$[\boldsymbol{\omega}_\times] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}.$$

The equation of the panels deflection angle α can be expressed by means of the usual Lagrange equation, adding the presence of a dissipation function D that acts as a natural damping for the system. In this way, the equation for the panels’ deflection can be written as:

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\alpha}} \right) - \left(\frac{\partial L}{\partial \alpha} \right) + \left(\frac{\partial D}{\partial \dot{\alpha}} \right) = 0$$

The Lagrangian function L is the difference between the kinetic energy T and the potential energy U ; these can be expressed for the spacecraft-wheels-panels system as:

$$\begin{aligned} 2T &= \int_R (\boldsymbol{\omega} \times \mathbf{r}) \cdot (\boldsymbol{\omega} \times \mathbf{r}) dm + \sum_{i=1}^4 \int_W (\boldsymbol{\omega} \times \mathbf{r} + \boldsymbol{\omega}_{wi} \times \mathbf{r}_{wi}) \cdot \\ &\quad (\boldsymbol{\omega} \times \mathbf{r} + \boldsymbol{\omega}_{wi} \times \mathbf{r}_{wi}) dm + m(\mathbf{V}_1 \cdot \mathbf{V}_1 + \mathbf{V}_2 \cdot \mathbf{V}_2) + 2I_p \dot{\gamma}^2 \\ \Rightarrow T &= \frac{1}{2} \boldsymbol{\omega}^T \mathbf{I} \boldsymbol{\omega} + \sum_{i=1}^4 \frac{1}{2} I_s \boldsymbol{\omega}_{wi}^2 + \sum_{i=1}^4 I_s \boldsymbol{\omega}_{wi} \boldsymbol{\omega} \cdot \hat{\mathbf{a}}_i + \frac{1}{2} m(\mathbf{V}_1 \cdot \mathbf{V}_1 + \mathbf{V}_2 \cdot \mathbf{V}_2) \\ &\quad + I_p \dot{\gamma}^2 \\ U &= \frac{1}{2} K(l^2 \alpha^2 + l^2 \alpha^2) = K l^2 \alpha^2 \end{aligned}$$

R indicates an integration over the rigid body, W an integration over the wheels; $\boldsymbol{\omega}$ is the angular rate vector in body coordinates, \mathbf{r} the radius of integration over the rigid body. $\boldsymbol{\omega}_{wi}$ represents the i -th wheel angular rate, \mathbf{r}_{wi} the radius of integration over the i -th wheel. \mathbf{V}_1 and \mathbf{V}_2 are the linear velocities of the two masses, \mathbf{I} is the matrix of inertia of the rigid body in body coordinates, excluding the presence of the appendages; the considered body frame is assumed to be the principal frame of inertia, and this implies \mathbf{I} to be diagonal. The assumption is likely in many cases, since most spacecrafts tend to be almost symmetric along two axes. I_p is the beam-like panel moment of inertia and I_s the wheel polar moment of inertia. $\hat{\mathbf{a}}_i$ is the unit vector that expresses the orientation of the i -th wheel axis in the body frame; later in this discussion, the notation a_{ix} , a_{iy} , a_{iz} will be used to indicate the components of the unit vector $\hat{\mathbf{a}}_i$ along the body axes. K is the flexional rigidity of the panels.

In the simplest model, the two solar arrays can be schematized as cantilever beams of length l , with tip mass m and a spring of flexional rigidity K , as in Fig. 3.5.

The angle α measures the panel deflection that is assumed to be equal in absolute value between the two arrays. The panels are considered to lie in a plane inclined of an angle γ with respect to the $X_{sat}-Y_{sat}$ plane: if γ is zero, the array is in the $X_{sat}-Y_{sat}$ plane; if γ is 90° , the array is in the $Y_{sat}-Z_{sat}$

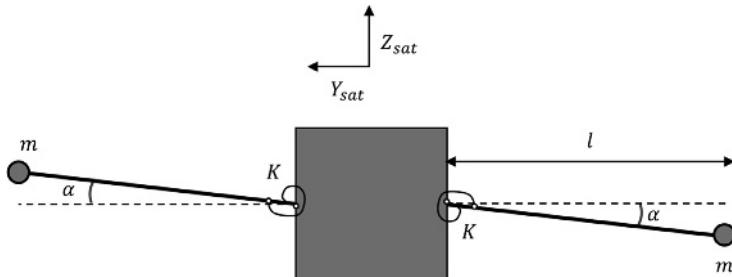


Figure 3.5 Flexible spacecraft scheme.

plane. It is reasonable to assume that the deflections are so small that $\sin \alpha \approx \alpha$, $\cos \alpha \approx 1$; therefore, the positions of the two masses can be identified by:

$$\mathbf{p}_1 = \begin{bmatrix} -dx + l\alpha \sin \gamma \\ -dy - l \\ -dz + l\alpha \cos \gamma \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} -dx - l\alpha \sin \gamma \\ dy + l \\ -dz - l\alpha \cos \gamma \end{bmatrix}$$

being dx , dy , and dz the distances in absolute value from the point of attack of the panel to the rigid body, to the total CoM, respectively, along the X_{sat} , Y_{sat} , and Z_{sat} axis. From this, it is possible to compute all the needed terms in the equation of the Lagrangian, leading to a final system in the form:

$$\mathbf{I} \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

where \mathbf{I} is the inertia matrix of the whole system:

$$\mathbf{I} = \begin{bmatrix} I_1 - \frac{I_l^2 \cos^2 \gamma}{2ml^2} & I_l \alpha \sin \gamma & \frac{I_l^2 \cos \gamma \sin \gamma}{2ml^2} - 2mdxdz \\ I_l \alpha \sin \gamma & I_2 & I_l \alpha \cos \gamma \\ \frac{I_l^2 \cos \gamma \sin \gamma}{2ml^2} - 2mdxdz & I_l \alpha \cos \gamma & I_3 - \frac{I_l^2 \sin^2 \gamma}{2ml^2} \end{bmatrix}$$

It is worth noticing that the matrix of inertia \mathbf{I} of the whole system is no longer diagonal because the presence of the arrays creates a misalignment between the body frame and the principal axis frame of reference. The variations caused to \mathbf{I} depend on α , and therefore the corresponding terms are

time-variant. Anyway, these terms are of at least three orders of magnitude inferior to the in-diagonal terms.

Example of a distributed parameters modeling

In this paragraph, the approach presented in Ref. [36] is described, where, for a higher representativeness of the model, a two-dimensional (2D) schematization of the array as a thin plate is chosen. In fact, in some cases, the assumption to reduce the solar array to a beam can be imprecise, given the real dimensions and geometry of the appendages. It is important to underline that the in-plane behavior of the plates is supposed to be negligible in the problem, which is absolutely the case for most space crafts since torsional in-plane rotations of the solar panels are far less likely to induce disturbances on the satellite attitude dynamics. Thus, the only possible responses of the structures are bending and twisting in the out-of-plane direction, under the hypothesis of the Kirchhoff kinematic model [29,30].

Also in this case, the Lagrangian formalism in quasicoordinates is used to find the equations of motion, but considering the alternative schematization of the solar panels, this time it shall be considered the position of a generic point P_1 on panel 1 and P_2 on panel 2, referring to Fig. 3.6:

$$\mathbf{r}_{P_1} = \begin{Bmatrix} R_h + x_{a_1} \\ \gamma_{a_1} + w_1(x_{a_1}, z_{a_1}, t) \\ z_{a_1} \end{Bmatrix} \quad \mathbf{r}_{P_2} = \begin{Bmatrix} -R_h + x_{a_2} \\ -\gamma_{a_2} - w_2(x_{a_2}, z_{a_2}, t) \\ z_{a_1} \end{Bmatrix}$$

In this way, the kinetic and the potential energy of the system can be found, but flexibility is still represented by continuous variable in space and time domains: the Lagrangian formalism can be applied only to discrete systems, so a suitable approximation is needed. In this way, the continuous variable w is replaced by a set of N generalized (or physical) coordinates, continuous in time domain but discretized in space. The discretization of a plate by means of a Ritz–Galerkin approximation, conceptually, divides the plate in a grid of beams. In a cantilever plate, there will be a set of cantilever beams (x-axis) and a set of free–free beam (z-axis). Each group is represented as a linear combination of a set of admissible functions \mathbf{N} (in space domain) times a set of generalized coordinates $\mathbf{u}(t)$:

$$w(x, z, t) = \sum_{i=1}^{N_c} \sum_{j=1}^{N_f} N_i N_j \mathbf{u}(t) = \mathbf{N} \mathbf{u}(t)$$

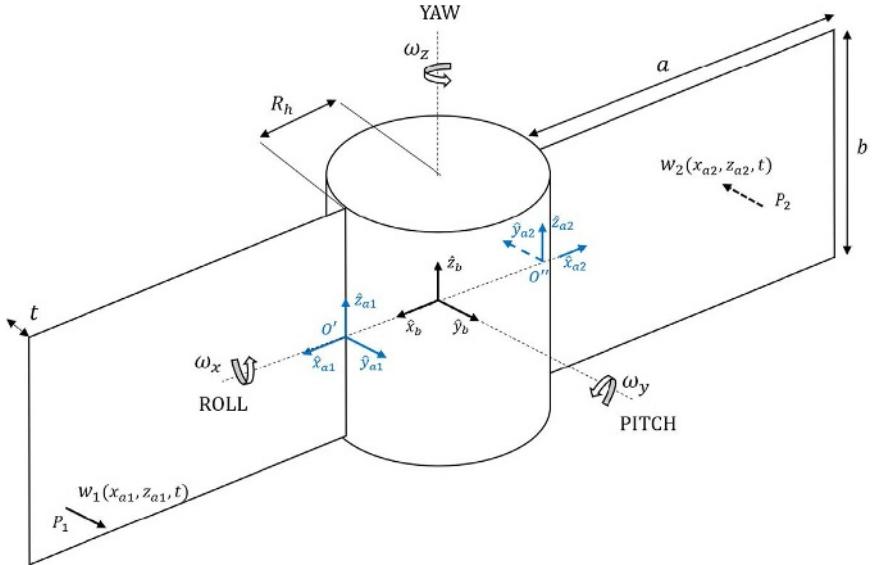


Figure 3.6 Schematics of the central hub and the two solar arrays, with main quantities used in computation.

The order of accuracy of the approximation depends on the grade and number of the functions. Using this approach, it is possible to get a system in the form of:

$$\mathbf{M}_{sys} \begin{Bmatrix} \dot{\boldsymbol{\omega}} \\ \ddot{\mathbf{u}}_1 \\ \ddot{\mathbf{u}}_2 \end{Bmatrix} + \mathbf{K}_{sys} \begin{Bmatrix} \boldsymbol{\omega} \\ \mathbf{u}_1 \\ \mathbf{u}_2 \end{Bmatrix} = \mathbf{t}$$

where \mathbf{M}_{sys} is the inertia matrix of the system, \mathbf{K}_{sys} is the stiffness matrix of the system, \mathbf{t} is the torque vector, $\boldsymbol{\omega}$ is the angular rate of the spacecraft, and \mathbf{u} are the displacements of the considered points of the arrays. The structure of the equations is the expected one, indeed, it is a set of $3 + 2N_cN_f$ Coupled Ordinary Differential Equations in a Continuous Time Domain, with mass and stiffness matrices counting for $(3 + 2N_cN_f) \times (3 + 2N_cN_f)$ elements. This is not acceptable considering reasonable computational capabilities; therefore, a typical approach is to introduce Modal Coordinates to reduce the size of the problem. The modes of the plates can be computed by solving the eigenvalues problem:

$$(\mathbf{M}_i - \lambda \mathbf{K}_i) \mathbf{v} = 0$$

The solution of the eigenvalues problem yields to a set of $N_c N_f$ mode shapes, each one vibrating at a specific discrete frequency ω_{ni} . Each mode, mathematically, represents a single independent harmonic oscillator vibrating at a specific frequency. The overall vibration of the plate will be a linear combination of these modes, $\mathbf{u}(t) = \mathbf{U} \mathbf{q}(t)$, where $\mathbf{q}(t)$ is the vector of the so-called $\mathbf{q}_i(t)$ Modal Coordinates, representing the weight in time of the i -th mode shapes, contained in the modal matrix \mathbf{U} . Depending on the frequency content of the physical loads in the problem, not all the modes will be equally excited: higher frequency modes typically belong to low mass participation factors and therefore can be discarded. How many modes should be included in the model is a trade-off between how much accuracy is lost in the truncation and the computational cost that can be sustained. Introducing also the damping elements, the final form of the system is obtained as:

$$\overline{\mathbf{M}}_{sys} \begin{Bmatrix} \dot{\omega} \\ \ddot{q}_1 \\ \ddot{q}_2 \end{Bmatrix} + \overline{\mathbf{C}}_{sys} \begin{Bmatrix} \omega \\ \dot{q}_1 \\ \ddot{q}_2 \end{Bmatrix} + \overline{\mathbf{K}}_{sys} \begin{Bmatrix} \omega \\ q_1 \\ q_2 \end{Bmatrix} = \bar{\mathbf{t}}$$

where $\overline{\mathbf{C}}_{sys}$ is the damping matrix of the system and the line above the matrices indicates the projection in the modal space.

Effects on dynamics and GNC

When the simple rigid body is considered, the simplified dynamics of the satellite can be represented by:

$$\mathbf{I}\dot{\boldsymbol{\omega}} = \mathbf{t}$$

being \mathbf{t} the torque vector, \mathbf{I} the satellite inertia matrix, and $\boldsymbol{\omega}$ the angular rate. The transfer function of this system is simply:

$$G(s) = \frac{1}{\mathbf{I} s^2}$$

which, since it has two poles in the origin, is represented in the Bode diagram by a line with a slope of $-40\text{dB}/\text{decade}$ for amplitude and a constant -180° in phase, as shown in Fig. 3.7.

The introduction of flexibility can be associated to the addition of two poles and two zeros for each vibrational mode, so that the transfer function becomes:

$$G(s) = \frac{s^2 + 2\xi\omega_\phi s + \omega_\phi}{\mathbf{I}s^2(s^2 + 2\xi\omega_n s + \omega_n)}$$

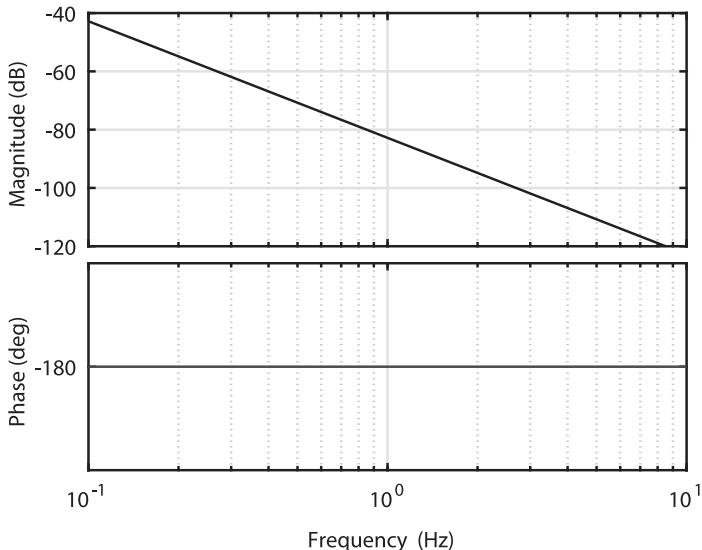


Figure 3.7 Bode diagram of a plant considering the rigid body model, with inertia of 350 kg m^2 .

with ξ the damping coefficient, ω_n the natural pulse of the system, and ω_ϕ the resonance of the spring. Therefore, there will be two breakouts in the Bode diagram: the first one is due to the two zeros and leads to a change of amplitude slope from $-40\text{dB}/\text{decade}$ to 0 and to a change in phase from -180° to 0; the second one is due to the two poles, and it brings back the amplitude to $-40\text{dB}/\text{decade}$ and the phase to -180° . Visually, two peaks can be identified for each modeled mode of vibration, corresponding to the resonance and the antiresonance of each mode, as can be seen in Fig. 3.8.

Clearly, GNC design should take care that the bandwidth of the control system shall remain well below the first frequency of resonance of the flexible modes (i.e., the frequency range in which the magnitude of the control action is greater than a certain minimum value shall be smaller and well separated from the resonance frequency of the flexible vibrations). On the other side, the resonance frequencies of the flexible modes shall not be too close to the controller frequency or its multiples, at least for the modes with most relevant mass participation factors [29,30].

The effect of flexibility is clearly visible during slews that excite bending modes: supposing a bang-bang attitude maneuver, the effect will be a first excitation of the panel to bending, then a second vibration when the torque changes in sign, and a last one when the torque comes back to 0. The second

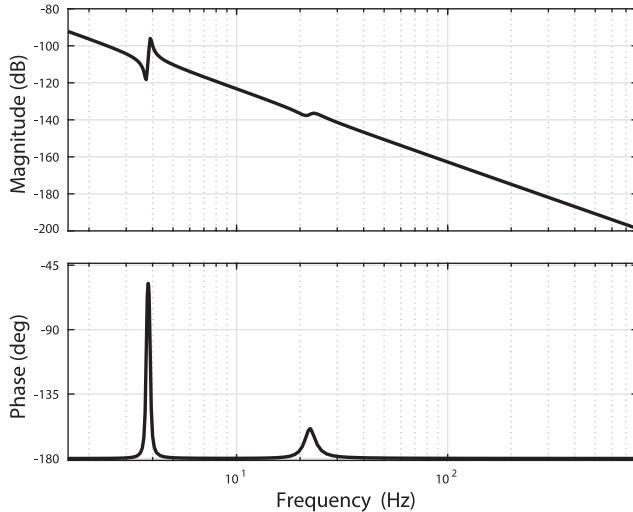


Figure 3.8 Bode diagram of a plant with inertia of 350 kg m^2 and the flexibility of the solar arrays. The first two bending modes have been included in the model: the first is clearly visible at $\omega_n = 4 \text{ Hz}$, the second one is much smoother at $\omega_n > 10 \text{ Hz}$. The peaks correspond to the two vibration frequencies of the solar panel.

excitation will be higher, since the torque step will be double than in the other two cases, as can be seen in Fig. 3.9. It is evident that these vibrations can affect attitude pointing both during the slew, but even after the maneuver, when a stable attitude is very likely to be needed to perform acquisitions

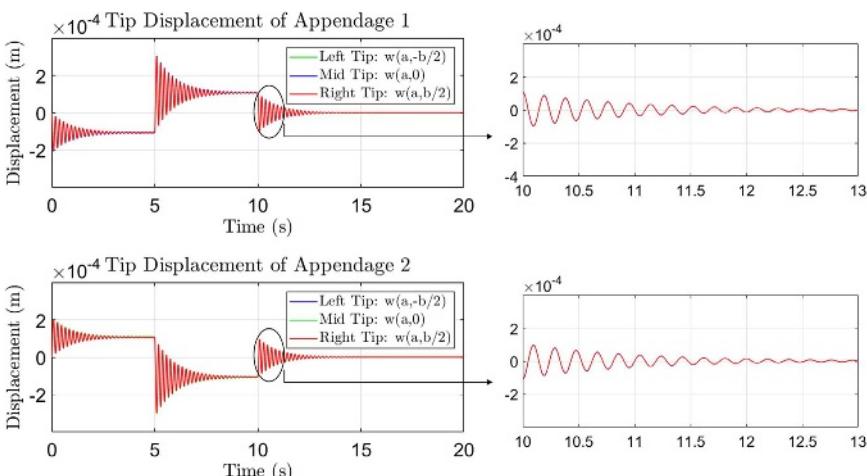


Figure 3.9 Solar array tip mass displacement during an attitude maneuver exciting bending mode.

of data. Therefore, depending on amplitude and duration of vibration versus pointing requirements, a tranquilization period could be needed before starting operations to allow stability of attitude pointing. On the other hand, the amplitude of the vibration and the time needed for vibrations to cease depend on the physical properties of the flexible appendage; therefore, appendage design, when possible, can be constrained to allow lower impact on attitude pointing and stability.

Sloshing

The fuel and liquids contained in the tanks inside the spacecraft shows a dynamical behavior that clearly cannot be assimilated to that of a rigid body, and consequently introduces oscillation effects into the spacecraft dynamics. The fuel motion within a tank is usually addressed to as *sloshing* [37,38].

When dealing with free liquids inside a tank, the classical and simplest approach to assess the impact of propellant sloshing on S/C dynamics and associated GNC performance consists in including reduced-order linear lateral sloshing models, such as pendulum or mass-spring mechanical models, as in the Abramson model [39], into the S/C dynamics model. This enables synthesis and first-order analysis of the control system in time- and frequency-domain at reduced computational costs, exactly the same way that has been analyzed in previous chapter about flexibility. This is done because equations of motions of oscillating point masses and rigid bodies are easier to be integrated in the equations of motion of a flexible spacecraft than those of a continuous liquid. By lateral sloshing is meant the standing wave formed on the surface of a liquid when a tank partially filled with liquid is oscillated. In the moment when the propellant mass impacts the wall of the fuel tank, it will transfer momentum to the spacecraft. If the tank is not at the center of mass of the vehicle, this will create an impulse moment that will affect the vehicle attitude. However, the pendulum or spring-mass models are adequate under gravity or gravity-like conditions, where inertial forces dominate over capillary forces in determining the fluid dynamics, which is true on ground or in diaphragm tanks, during thrusters firing or during attitude acceleration due to slews. When dealing with Propellant Management Device tanks under microgravity conditions, the simple pendulum mechanical models is no more accurate enough, since gas and fuel are not separated. In these cases, the propellant tends to spread over all the tank available internal surface, leading bubbles of gas to move toward the center of the tank, thus creating a configuration whose dynamics needs

specific computational fluid dynamics (CFD) analyses to be computed. To assess whether the motion is dominated or not by inertia forces, two nondimensional numbers can be taken into consideration when the fluid is in motion: the first one as the ratio between inertia and capillarity forces, the *Weber* number; the other one is the *Froude* number that compares inertia to gravity actions.

$$We = \frac{\rho v^2 L}{\sigma}$$

$$Fr = \frac{v^2}{gL}$$

being ρ the density of the fuel, v its velocity, L the characteristic dimension, σ the surface tension, and g the gravity acceleration. When these nondimensional numbers are much greater than 1, then the fluid motion is driven by inertia forces and lateral sloshing can be considered not negligible. In this case, mechanical models are accurate enough and CFD analyses can be avoided.

Following the largely used mass-spring representation of sloshing, the general form of the equations of motion for each tank slosh mode frequency is the one of a small point mass attached to an extended, rigid, center body, allowed to execute small oscillations about its equilibrium positions. The equations of motion may be derived from a Lagrangian formulation, following the same procedure described in the previous paragraph when dealing with general flexibility. The effects on pointing and stability are the same, reminding that sloshing only is effective where relevant accelerations are present in the system; thus, it can be usually neglected during slow rate data acquisitions. However, if extremely accurate analyses are necessary for these specific conditions, CFD analyses of the fluid motion inside the tanks may be needed.

The slosh motion is characterized by a natural frequency that is a function of the tank shape and of the axial acceleration imposed on the tank; for a spherical tank, this becomes:

$$\omega_n = \frac{P_\omega}{\sqrt{r/g}}$$

where P_ω is the natural frequency parameter, r the tank radius, and g the acceleration the tank is subjected to. The acceleration influencing the sloshing dynamics, g , is the dynamically induced acceleration acting along

the axis of the tank; for practical applications, it can be selected to be equal to the average gravitational acceleration along the orbital motion. P_ω is dependent from the geometry of the tank and from the properties of the fluid [39,37,38]. Note that the sloshing model shall also consider the liquid damping. For surface-tension tank technology, the slosh motion damping is primarily provided by the liquid viscosity; for diaphragm tank technology, the damping ratio is much higher than for surface tension tanks due to the elastic flexing of the diaphragm and to the increased viscous effects at the liquid-diaphragm interface. The liquid damping is then included in the dynamical models with the equations of a mechanical damper, and the sloshing equivalent mechanical model results in a damped pendulum or mass-spring-damper system [29]. Similar to the flexibility, the sloshing is commonly directly integrated in the equations of motion, which are then combined with forces and torques of other perturbation sources.

Parasitic forces and torques during thrusters firing

Spacecrafts often need a propulsion system to perform orbital maneuvers: for example, a satellite in LEO can use propulsion for initial orbit acquisition or orbit maintenance, or even to perform collision avoidance maneuvers. Propulsion is used also as the most common solution for deorbiting a spacecraft at the end of its lifecycle to comply with the current international guidelines to limit orbital debris. Similarly, geostationary satellites typically use thrusters for station-keeping and a large apogee motor for the initial acquisition of their nominal orbit after the injection in a geostationary transfer orbit by the launcher. Whatever the purpose of the on-board propulsion system, the system shall be designed taking into account the fact that propulsive forces are just ideally in the designed direction: in reality, those directions will have a misalignment with respect to the nominal ones, creating an out-of-plane component of the force that also generates a torque. Moreover, the position of the CoM can have a displacement with respect to the thrust direction, and it will also vary during the spacecraft lifetime as the propellant is consumed, leading to further torques generated about the CoM. In addition, if the propulsion system is based on thrusters used symmetrically with respect to the CoM, so that torques are not nominally created, then it has to be considered that the real amount of thrust generated by each thruster will differ from the nominal one. Typically, in the order of 5%–10% up to 15%. This results in a worst case in which one thruster provides less force than it is supposed to, and its symmetric one delivers more than expected. This of course results in undesired force and torque

contributions around the CoM. The following discussion will aim to the parasitic torque effects, since the force ones are inherently described to compute the torques.

As long as thrusters are continuously firing, these parasitic forces and torques manifest as constant force and torque disturbances. Clearly, these disturbances are only present during firing, therefore only during orbital maneuvers, if the propulsion system is aimed at that purpose. However, during these phases of the satellite lifecycle, every steady-state error will result in a constant misalignment between the required and the delivered ΔV . Therefore, it is of utmost importance to compensate for these constant disturbances by designing a controller that removes, or at least limits, any steady-state error in presence of constant disturbance forces and torques.

It is important to underline that parasitic torques, when present, are one of the highest attitude disturbances acting on the satellite. Obviously, errors and disturbances in thruster firings do not affect only the attitude motion, but the whole translational and rotations dynamics. As said, even if this section focuses on parasitic torques, the internal perturbation forces due to thrusters' errors directly apply to the translational GNC.

In the following, a short insight on how to compute the different contributions is provided.

Deviation angle

The real thrust direction will be different from the ideal one, by a deviation angle that is the sum of two contributions: the difference between the thrust direction and the longitudinal axis of the thrusters, and the difference between this longitudinal axis and the spacecraft reference direction (i.e., the direction where we desire our thruster to be along). These two contributions to the total deviation angle can vary in a range from 0.1° to 1° each, depending on the quality level of the thruster manufacturing and the attention during the spacecraft assembly phase. In fact, the most proper way to reduce the angle between the longitudinal axis of the thrusters and the spacecraft reference axis is to have an alignment measure of that angle and then act to reduce it with mechanisms on the thruster, if they are present. The measurement is typically a laser measure, acquiring the output circumference of the nozzle to define the perpendicular to that surface, to be compared with the spacecraft reference axes.

If we assume to have two symmetrical thrusters with respect to a reference axis of the satellite, as in Fig. 3.10, where a simplified 2D schematic is provided, the worst case for the generation of parasitic torques is when both

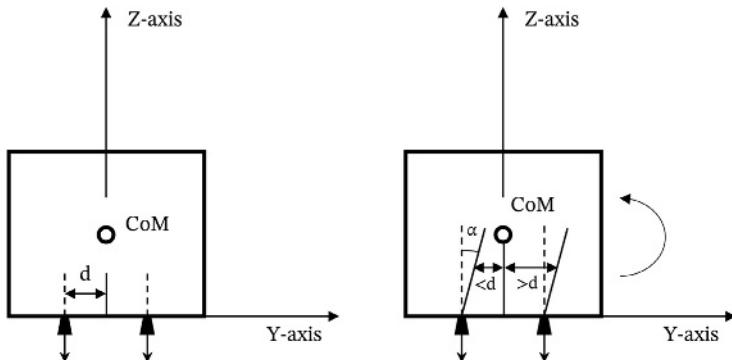


Figure 3.10 Parasitic torque generated by a deviation angle of the thrusters.

thrusters have a deviation angle in the same direction, so that the difference between arms is maximum. Both contributions to the deviation angle discussed before are independently affecting the system.

In this case, a torque around the x -axis will be generated, approximated by:

$$t = 2 F \sin \alpha$$

where t is the generated parasitic torque, F is the nominal force of each thruster, α is the deviation angle, and h is the CoM height with respect to force application point.

Center of mass variation

With reference to Fig. 3.10, the same concept of generating a parasitic torque applies when it is not the thrust direction deviating, but the CoM moving from the longitudinal axis. For example, if the CoM is biased from the z-axis along the y-axis, then, one thrust will have a longer arm than the other, creating a parasitic torque:

$$t = 2F\Delta y$$

where Δy is the offset of the CoM along the y-axis.

Therefore, it is important to configure the spacecraft such that the displacement of the CoM from the axis parallel to thrust direction is contained within an acceptable threshold, i.e., a threshold that allows for manageable parasitic torques. However, CoM position can also vary throughout the mission, for example, due to fuel consumption inside the tank: in this case, care must be paid in order to let the CoM move possibly only along the longitudinal direction, i.e., the z-axis in the example in

Fig. 3.10, so that nominally no parasitic torque is created. In a real case, a slight parasitic torque will be created even in that case, due to the presence of misalignments and the coupling between an even small deviation angle and the CoM movement along Z.

Thrust magnitude accuracy

The third contribution to parasitic torques is provided by the fact that thrusters deliver a force with an associated accuracy: in fact, given the same conditions (i.e., temperature of the tank and pressure), a thruster can provide a certain percentage more, or less, than the nominal thrust for those conditions. In other words, the delivered thrust will be: nominal $\pm X\%$, where X is typically less than 15% at 3σ . This translates in another contribution to the total parasitic torque. As already stated, the worst case occurs when one thruster delivers less torque than it was expected, while its symmetric delivers more, leading to:

$$t = 2\Delta F d$$

where ΔF is the delta-thrust respect to the nominal case and d the arm between the thrusters.

Effects on dynamics and GNC

Once all the distinct contributions have been assessed, the total parasitic torque can be computed. It is a good practice to not directly sum the three terms, as those are worst cases that are very unlikely to occur at the same time, but to use instead a root-sum-square to define the total disturbance torque in a less pessimistic, yet very conservative way.

The impact of parasitic torques upon the spacecraft is twofold: on one side, GNC designers shall ensure that there is sufficient control authority to cope with the disturbance torque, and at the same time with all the other disturbances acting on the spacecraft. On the other side, if momentum exchange actuators (e.g., reaction wheels) are used on the satellite as the primary controlling equipment, it shall be assessed whether the momentum build-up caused by parasitic torques during thrusters firing can be coped with the available momentum storage given the size of the wheels and the configuration of the wheel assembly. Indeed, if a 0.1Nm parasitic torque is generated during firing, then a 100s burn will cause 10Nm in momentum build-up that shall be stored in reaction wheels. It is not rare that one of these two assessments, or even both, leads to conclude that the boarded reaction wheels cannot cope with the generated parasitic torques. In case it is not

possible to board larger wheels, there are essentially two ways to solve the issue: limiting the parasitic torque, for example, reducing the deviation angle or the CoM displacement or controlling attitude during firing by off-modulating thrusters forces instead of using reaction wheels, or using a combination of both methods, as described in Ref. [40]. In the cases where only the momentum build-up may lead to GNC issues, while control authority is sufficient to grant controllability, it could be considered to limit the firing duration and split orbital maneuvers in more thruster burns. However, this option is less preferable given the evident impact on the operability of the spacecraft.

Electromagnetic disturbances

Earth magnetic field has a negligible effect on the translational orbital motion, but it strongly interacts with spacecraft residual dipole, generating a disturbance torque equal to:

$$\mathbf{t}_{\text{mag}} = \mathbf{m} \times \mathbf{B} \quad (3.15)$$

where \mathbf{m} is the satellite residual dipole and \mathbf{B} is the Earth's magnetic field. Both these quantities need to be evaluated in order to assess this disturbance torque. Earth's magnetic field can be modeled as described above in the external space environment sections, and its evaluation is relatively simple by exploiting the standard reference models.

The same cannot be said for the other term of the equation, the spacecraft residual dipole. This can be evaluated in essentially two ways: a detailed budget of the residual dipole of each component of the spacecraft or a measurement of the spacecraft magnetic field. The first is simpler and less expensive but assumes that information is available for each relevant component of the satellite, typically payloads, electronic drives, magnetic parts, etc. Evaluation of the dipole can otherwise be done indirectly through a measurement of the spacecraft magnetic field, possibly from several positions and at different distances, and then applying the formula of the magnetic field produced by a dipole at a given distance:

$$\mathbf{B} = \frac{\mu_0}{4\pi} \left[\frac{3\mathbf{r}(\mathbf{m} \cdot \mathbf{r})}{|r|^5} - \frac{\mathbf{m}}{|r|^3} \right]$$

where μ_0 is the magnetic permeability in vacuum. The formula cannot be simply reversed, so a dipole model could be fitted to the magnetic field measurements using a nonlinear least squares optimization, as performed in

Ref. [41]. When performing such measurements, extreme care must be paid in order to reduce the effect of external magnetic sources and the magnetic field due to sources different from the spacecraft itself: for this reason, the utilization of an anechoic chamber is addressed as the most proper solution for this kind of test. As such, this kind of measurement is typically used for smaller satellites when estimation is not possible or not sufficiently accurate. For example, when commercial off-the-shelf components are used, the residual dipole information of the equipment is usually not available. On the contrary, for larger satellites, which can barely fit an anechoic chamber and for which information are likely available from suppliers, a dipole budget is typically assessed. As an order of magnitude, a satellite weighing between 500 and 800 kg can have a residual dipole in the order of a few Am^2 (i.e., 1 to even 4–5) that decreases for smaller satellites, typically ranging below 0.1–0.01 Am^2 for CubeSats [42].

The effect of this disturbance is certainly related to the generation of a torque according to Eq. (3.15) that shall be compensated by the attitude control system. However, there is another subtle effect: residual magnetic dipoles can degrade magnetometers measurements, since the spacecraft dipole generates a magnetic field that locally distorts Earth's field. The strength of this local field decreases with the cube of the distance from the source of the dipole, thus it is important to accommodate magnetometers inside the satellite far from any sources of disturbance, as far as it is enough to not degrade magnetic field measurements. If magnetic torquers are used on the spacecraft, it is a common practice both to place magnetometers as far as possible from the torquers and to acquire magnetometers data in a duty cycle only when torquers are off.

Internal vibrations

A spacecraft can be affected by several sources of vibrations, that in general can be associated to all the rotary parts present on board: reaction wheels, Solar Array Drive Mechanism, rotating motors in the payload, etc. These components emit microvibrations as forces and torques, whose amplitude and frequency depend on the equipment speed of rotation. Moreover, these forces and torques are modulated by the satellite structure and ultimately result in perturbations on the spacecraft dynamics. Therefore, they affect the relative stability of the pointing error, and they are of particular importance for missions requiring high pointing accuracy and stability.

A general approach to study the problem is to develop a FEM model for each relevant component of the spacecraft, integrate it into the spacecraft

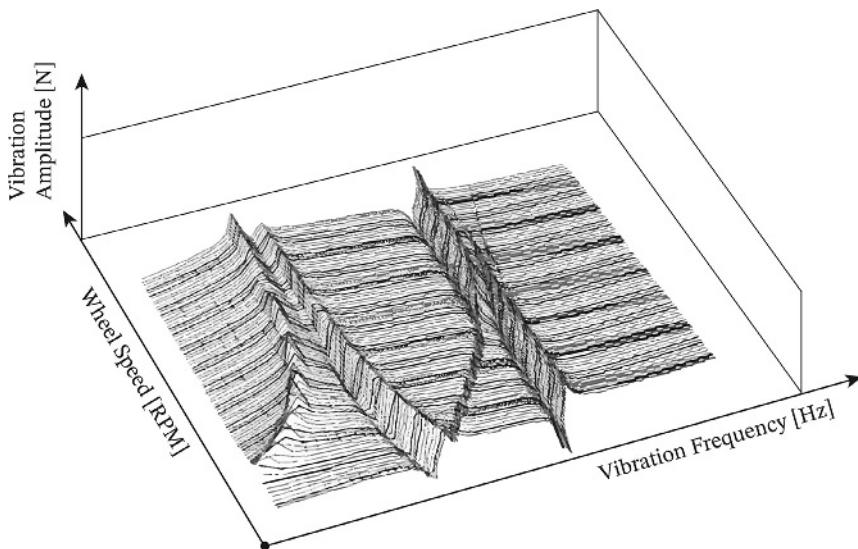


Figure 3.11 3D vibrations amplitude versus the speed of the wheel and vibrational frequency.

FEM model, and then analyze the overall response of the system to micro-vibrations in different situations. Since reaction wheels are in most cases one of the highest contributors to microvibrations, due to wheel imbalance, bearing torque noise due to imperfections, bearing friction effects, motor torque ripple, motor cogging, etc., particular care is taken while analyzing wheels behavior. A typical outcome of this analysis is the *waterfall plot* for the reaction wheels, a three-dimensional graph representing the amplitude of their vibrations versus the speed of the wheel and its vibrational frequency, as in Fig. 3.11 taken from Ref. [43].

This graph is useful to evaluate the frequencies of resonance of the wheel, to see if they can interact with the spacecraft structure, and how operative wheel speed can affect the amplitude of these critical frequencies. However, the end-result of the microvibration analysis is typically the effect of the microvibration on the line of sight of instruments present on board, as can be well seen in Ref. [44]. Microvibrations in space missions are a wide topic that cannot be treated in detail here, but the topic can be deepened in [43–45,46].

Reaction wheel jitter

If, on one side, forces and torques due to microvibrations can cause coupling effects with the spacecraft structure, leading to possible resonances and a

depointing of the payload, on the other side, a similar final depointing effect can be caused by torques generated by reaction wheels due to their intrinsic vibration, referred to as jitter. Jitter torques on the spacecraft are caused by static and dynamic unbalances of the reaction wheels, which generate forces, and thus torques, applied to the mounting structure of the wheels as in Fig. 3.12.

Static imbalances are radial asymmetries in mass distribution: each spinning wheel with a static imbalance will impose a periodic force on the spacecraft. The magnitude of the force vector is constant; however, its direction changes with time. As explained in Ref. [45], in the case of a wheel with spin axis aligned with the z-axis, the force on the spacecraft breaks down as:

$$\mathbf{f}_z = S_z \omega_z^2 \cos(\omega_z t + \phi_z) \hat{\mathbf{x}} + S_z \omega_z^2 \sin(\omega_z t + \phi_z) \hat{\mathbf{y}}$$

Being S_z the static unbalance (in kg m), ω_z the wheel speed in rad/s, and ϕ_z the phase, considered null here as a simplifying assumption. These forces are applied at the site of the reaction wheel assembly: as a result, any force not acting through the CoM also acts as a torque on the spacecraft. Below it is shown the set of all forces and torques on the body due to any static imbalance as a function of the wheel angular velocities:

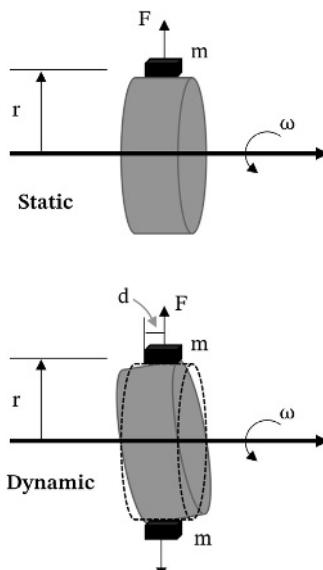


Figure 3.12 Reaction wheels jitter representation.

$$\begin{aligned}\mathbf{f}_s(\omega_x, \omega_y, \omega_z) &= \left(S_y \omega_y^2 \sin(\omega_y t + \phi_y) + S_z \omega_z^2 \cos(\omega_z t + \phi_z) \right) \hat{\mathbf{x}} \\ &+ \left(S_z \omega_z^2 \sin(\omega_z t + \phi_z) + S_x \omega_x^2 \cos(\omega_x t + \phi_x) \right) \hat{\mathbf{y}} \\ &+ \left(S_x \omega_x^2 \sin(\omega_x t + \phi_x) + S_y \omega_y^2 \cos(\omega_y t + \phi_y) \right) \hat{\mathbf{z}} \\ \mathbf{t}_s(\omega_x, \omega_y, \omega_z) &= \mathbf{r}_w \times \mathbf{f}_s(\omega_x, \omega_y, \omega_z)\end{aligned}$$

Dynamic imbalances are asymmetries in mass distribution across the thickness of the wheel. Combining the effects of three wheels of a typical reaction wheel assembly, the net torque on the spacecraft due to dynamic imbalances D_x , D_y , D_z (in [kg m^2]) is:

$$\begin{aligned}\mathbf{t}_d(\omega_x, \omega_y, \omega_z) &= \left(D_z \omega_z^2 \sin(\omega_z t + \phi'_z) + D_y \omega_y^2 \cos(\omega_y t + \phi'_y) \right) \hat{\mathbf{x}} \\ &+ \left(D_x \omega_x^2 \sin(\omega_x t + \phi'_x) + D_z \omega_z^2 \cos(\omega_z t + \phi'_z) \right) \hat{\mathbf{y}} \\ &+ \left(D_y \omega_y^2 \sin(\omega_y t + \phi'_y) + D_x \omega_x^2 \cos(\omega_x t + \phi'_x) \right) \hat{\mathbf{z}}\end{aligned}$$

It is worth to notice that since these are pure torques, the net force on the spacecraft is null.

Depending on wheel size and imbalances, satellite inertia, and wheel speed, jitter final effect on depointing the line of sight of the payload can be very relevant in some cases, typically those missions with large satellites requiring a high degree of pointing accuracy or results to be negligible in other cases. Jitter effect on the pointing error varies with the square of the reaction wheel speed, thus suggesting that, for a given wheel size and imbalance and satellite inertia, varying (i.e., reducing) the working point of the reaction wheels is highly impactful on this disturbance, and it can be a good ally to reduce undesired depointing effects in situations where jitter is relevant. Further details about reaction wheels disturbances will be discussed in the [Chapter 7 – Actuators](#).

Parasitic forces and torques due to plume impingement

Another cause of disturbances acting on the satellite is represented by thrusters' plume impingement, that is usually negligible, but can become relevant in particular situations.

In general, the flux exiting thrusters' nozzle is not linear but widespread, as in [Fig. 3.13](#).

This has two main consequences: on one hand, part of the flux can possibly reach solar panels and degrade their performance; on the other

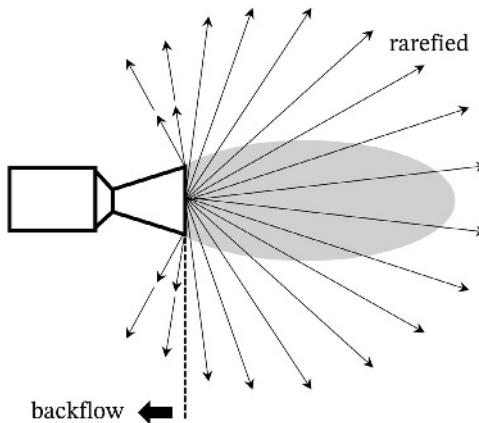


Figure 3.13 Thruster flux representation.

hand, every part of the flux which hits other parts of the satellite creates a force acting on the spacecraft, and thus a torque with arm, the segment between the application point of the force and the CoM of the satellite. Even if thrusters are symmetric, their plume forces could be asymmetric due to different alignments and different thrust magnitude accuracy. Typically, these parasitic forces and torques are negligible if compared to other sources of disturbance due to the fact that the accommodation of the thrusters is usually such that the exit surface of the nozzle is perpendicular and opposite to relevant surfaces of the spacecraft (e.g., solar arrays). However, there can be rare cases where the accommodation of thrusters and the configuration of large solar panels is such that relevant parasitic torques can be created during prolonged thrusters firing, as it was experienced for the ESA Sentinel 1 satellite [47]. For modeling purpose, please refer to Ref. [48].

Thermal snap

Thermal shock can happen during transition from sunlight to shadow or vice versa, when solar panels and other sensible components experience a fast, high thermal gradient. In fact, thermal conditions are significantly different for the surfaces exposed to the Sun compared to the back ones. This thermal difference between the two sides of the panels can possibly lead to thermal snaps, exciting vibrations in the panels, which in turn cause perturbations on the attitude dynamics. However, thermal snap shall be considered only for spacecraft with very large, flexible solar arrays or even booms, and for missions where fine attitude pointing during flight over the terminator line or in its proximity is important. Thus, this is typically not the case for optical satellites.



Internal perturbations modeling guidelines

International standards and guidelines do not enter the details of internal perturbations modeling because they are extremely mission- and system-dependent, and their generalization and standardization are extremely difficult.

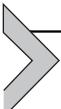
International standards classify the internal perturbations as error sources, and they give general guidelines on how to handle these internal error sources in the GNC design. Internal error sources are elementary physical processes originating from a well-defined internal source, which contributes to a GNC performance error. Obviously, sensor errors also fall in this category, but microvibrations, subsystem flexibility, internal mass movements, misalignments, thermal distortions, jitter, etc., are listed by the standards as relevant internal error sources.

The guidelines suggest handling these disturbances by listing all significant internal perturbations contributing to the GNC design [49,50]. Then, they shall be preliminarily quantified and ranked by disturbance magnitude. According to the required accuracy level of the system, and in agreement with the details of the external perturbations modeling, the least relevant internal perturbations may be discarded. A justification for neglecting some potential error sources should be maintained in the GNC design documentation to show that they have been considered. Error sources shall be ranked in terms of mean and standard deviation along each spacecraft axis. Furthermore, the internal perturbations shall be also classified according to their temporal distributions and reproducibility. In fact, they can be:

- Time constant.
 - Periodic.
 - Transient.
 - Linearly varying (i.e., increasing/decreasing).
- Moreover, their evolution can be:
- Deterministic.
 - Random.

In the case of random internal disturbances, they are associated to a specific distribution, which can be uniform, Gaussian, or another one. When the list of the selected relevant internal perturbations is available, with their magnitude, temporal evolution, reproducibility, and distribution, the GNC design can be performed, exploiting system simulations including the most updated internal perturbations' models.

It is relevant to remark that the accuracy level of external and internal perturbations modeling shall be balanced, including effects of comparable magnitude in the two domains. Moreover, as a general rule, we should follow a conservative approach and include as many error sources as possible rather than assuming that some are negligible. Even small errors can contribute significantly if there are enough of them.



What is relevant for GNC?

The space environment sets the boundary conditions for any spacecraft while on orbit along its operative lifetime. Evidently, it is strongly characterizing the entire design of a space mission. The same is true for the GNC system, which is particularly affected by the space environment in terms of perturbative forces and torques. Thus, the GNC design shall be based on a correct space environment modeling, in terms of accounted perturbations and evaluation of their effects on the spacecraft dynamics. As discussed in [Chapter 10 – Control](#), the disturbances on the dynamics enter the GNC loop and induce errors on the desired state, if not properly managed. Moreover, perturbations may lead to actuators ineffectiveness if the GNC design underestimate the environmental perturbations, both external and internal.

One of the main aspects to consider is what kind of perturbations need to be accounted to model the space environment. Indeed, as presented in this chapter, the perturbation sources have different impact on the spacecraft dynamics according to the region of space the spacecraft is operating in (e.g., central attractor, distance from the surface, inclination with respect to equatorial planes), to its internal components and system characteristics, to its dynamical state (e.g., velocity, rotation rates), to its current GNC mode (e.g., actuators in use, required accuracy and stability), just to name the principal ones. Thus, it is evident that a complete system level analysis shall be performed beforehand to define the operative external and internal environments. Then, a trade-off between desired accuracy and design complexity, also considering the associated computational and modeling costs, is performed to set the threshold when a particular perturbation source can be neglected. As already said, this analysis phase may be supported by preliminary models and tools providing a rough estimation of any present perturbation to rank them and conservatively select those that may truly affect the dynamics and the GNC design. Finally, the definitive perturbation models are implemented according to the international standards and

guidelines. These models shall be used to simulate and analyze the spacecraft in any of its operative conditions in order to drive the design of the space mission in all its elements.

Particular care must be given to internal perturbations and spacecraft configuration analyses, since they may prevent actuator and sensor ineffectiveness. To make some examples, large offsets in the CoM or in the inertia moments may lead to oversized actuators, bad electromagnetic configuration can deteriorate the nominal performance of magnetic sensors and actuators, wrong thermo-structural design might create dynamical coupling that are difficult to be managed and controlled.

Obviously, even if more standardized and easier to be analyzed, external environment shall not be overlooked. We shall always have in mind that there is no environment model or set of perturbation sources that fits for any space mission. A superficial environment analysis will lead to a failed GNC design. For instance, some sensors and actuators may not properly work on certain orbits (e.g., due to weak or unknown magnetic fields, due to unmodeled sunlight reflection or albedo from the planet). In general, a GNC design can be not adequate if relevant perturbation sources are neglected or mismodeled (e.g., overlooked torques exceeding actuation capabilities, underestimated environmental phenomena disturbing sensor measurements).

Perturbation sources need to be tuned at a level of accuracy that is coherent with the whole GNC design. The modeling details shall be coherent and comparable between external and internal perturbing terms. Moreover, as a rule of thumb, if a perturbation source is included, all the terms that are more relevant must be included. For example, considering a LEO orbiting spacecraft, the rough orders of magnitude of the external orbital perturbations are:

- Keplerian acceleration: $\sim 8.5 \text{ m/s}^2$.
- $J_2: \sim 10^{-2} \text{ m/s}^2$.
- $J_{22}: \sim 10^{-4} \text{ m/s}^2$.
- $J_{66}: \sim 10^{-5} \text{ m/s}^2$.
- Atmospheric drag: $\sim 10^{-5}/10^{-8} \text{ m/s}^2$.
- Moon/Sun: $\sim 10^{-6} \text{ m/s}^2$.
- Dynamic solid tides: $\sim 10^{-7} \text{ m/s}^2$.
- SRP: $\sim 10^{-7} \text{ m/s}^2$.

Hence, if SRP is considered, all the previously listed terms shall be included as well. Similarly, internal perturbations leading to accelerations larger than $\sim 10^{-7} \text{ m/s}^2$ shall be accounted.

Space environment models are not only used for GNC design and analyses, but they are also embedded in GNC system to have an on-board representation of the environment, which can be used for GNC purposes. For example, planetary ephemerides may be used to implement guidance targets or reference pointing directions, Sun or Earth models are used to be compared with sensor measured data in navigation functions, internal vibration or flexible models can be included in the control laws to have improved performance. Therefore, having a proper environment modeling has also a direct impact on the GNC functions. Mismodeled effects may lead to errors with respect to the desired reference states or wrong disturbance rejection actions. Analogous problems arise from poorly accurate models. However, the on-board models shall carefully balance accuracy with the required computational resources. In these regards, it is crucial to balance the fidelity of the environment models with the practical needs and with the requirements of the GNC system. Excessively accurate environment models waste the on-board resources without improving the final performance, on the contrary too rough models can limit the precision of a refined GNC chain.

References

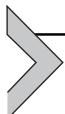
- [1] I. Newton, *Philosophiae Naturalis Principia Mathematica* 2, 1833 typis A. et JM Duncan.
- [2] A.M. Legendre, *Recherches sur l'attraction des sphéroïdes homogènes*, De l'Imprimerie Royale, 1785.
- [3] E.W. Hobson, *The Theory of Spherical and Ellipsoidal Harmonics*, Cambridge University Press, 1931.
- [4] W.M. Kaula, *Theory of Satellite Geodesy*, Chap. 1, Blaisdell Publishing Company, Waltham, Massachusetts, 1966.
- [5] A.E.H. Love, The yielding of the Earth to disturbing forces, in: *Proceedings of the Royal Society of London - Series A: Containing Papers of a Mathematical and Physical Character* 82, 1909, pp. 73–88.
- [6] D.D. McCarthy, G. Petit, *IERS conventions (2003)*. International Earth Rotation And Reference Systems Service (IERS)(Germany), *Journal of Geodesy* 77 (10) (2004) 585–678.
- [7] E.C.S. Secretariat, Space Engineering: Space Environment (ECSS-E-ST-10-04C), European Cooperation for Space Standardization, 2020. <https://ecss.nl/>.
- [8] <https://www.ngdc.noaa.gov/IAGA/vmod/igrf.html>.
- [9] G. Batchelor, *An Introduction to Fluid Dynamics* (Cambridge Mathematical Library), Cambridge: Cambridge University Press, 2000, <https://doi.org/10.1017/CBO9780511800955>.

- [10] N. Sissenwine, M. Dubin, H. Wexler, The US standard atmosphere, 1962, *Journal of Geophysical Research* 67 (9) (1962) 3627–3630.
- [11] D. Rees, J.J. Barnett, K. Labitzke, COSPAR International Reference Atmosphere, Pt. 2: middle atmosphere models, *Advances in Space Research* 10 (12) (1986) 1990.
- [12] L.G. Jacchia, Revised Static Models of the Thermosphere and Exosphere with Empirical Temperature Profiles, SAO Special Report, 1971, p. 332.
- [13] L.G. Jacchia, Thermospheric Temperature, Density, and Composition: New Models, SAO Special Report, 1977, p. 375.
- [14] D.A. Vallado, Fundamentals of Astrodynamics and Applications, vol. 12, Springer Science & Business Media, 2001.
- [15] <https://www.spacex.com/updates/> 8th February 2022.
- [16] P.C. Knocke, J.C. Ries, B.D. Tapley, Earth Radiation Pressure Effects on Satellites, Proceedings of the AIAA/AAS Astrodynamics Conference, Washington DC, 1988, pp. 577–586.
- [17] A. Roy, Orbital Motion, John Wiley & Sons, New York, 1988.
- [18] F.T. Geyling, H.R. Westerman, Introduction to Orbital Mechanics, Addison-Wesley Aerospace Series, 1971.
- [19] A.C. Long, J.O. Cappellari Jr., C.E. Velez, A.J. Fuchs, Goddard Trajectory Determination System (GTDS) Mathematical Theory, National Aeronautics and Space Administration/Goddard Space Flight Center, 1989.
- [20] The Astronomical Almanac, UKHO General Publication: GP 100-21, 2021.
- [21] J. Meeus, Astronomical Algorithms, Willmann-Bell, 1991.
- [22] J.C. Mason, D.C. Handscomb, Chebyshev Polynomials, Chapman and Hall/CRC, 2002.
- [23] I.I. Alexeev, V.V. Kalegaev, E.S. Belenkaya, S.Y. Bobrovnikov, Y.I. Feldstein, L.I. Gromova, Dynamic model of the magnetosphere: case study for January 9–12, 1997, *Journal of Geophysical Research: Space Physics* 106 (A11) (2001) 25683–25693.
- [24] N.A. Tsyganenko, D.P. Stern, Modeling the global magnetic field of the large-scale Birkeland current systems, *Journal of Geophysical Research: Space Physics* 101 (A12) (1996) 27187–27198.
- [25] J.M. Picone, A.E. Hedin, D.P. Drob, A.C. Aikin, NRLMSISE-00 empirical model of the atmosphere: statistical comparisons and scientific issues, *Journal of Geophysical Research: Space Physics* 107 (A12) (2002). SIA-15.
- [26] B.R. Bowman, W.K. Tobiska, F.A. Marcos, C. Valladares, The JB2006 empirical thermospheric density model, *Journal of Atmospheric and Solar-Terrestrial Physics* 70 (5) (2008) 774–793.
- [27] B. Bowman, W.K. Tobiska, F. Marcos, C. Huang, C. Lin, W. Burke, A new empirical thermospheric density model JB2008 using new solar and geomagnetic indices, in: AIAA/AAS Astrodynamics Specialist Conference and Exhibit, 2008, p. 6438.
- [28] W.M. Folkner, J.G. Williams, D.H. Boggs, R.S. Park, P. Kuchynka, The planetary and lunar ephemerides DE430 and DE431, *Interplanetary Network Progress Report* 196 (1) (2014).
- [29] L. Meirovitch, Fundamentals of Vibrations, Waveland Press, Long Grove, 2010.
- [30] J.L. Junkins, Introduction to Dynamics and Control of Flexible Structures, AIAA Education Series, Reston, 1993.
- [31] R. Zhang, S.N. Singh, Adaptive Output Feedback Control of Spacecraft with Flexible Appendages, American Control Conference, 2001.
- [32] G. Bodineau, S. Boulade, B. Frapard, W. Chen, S. Salehi, F. Ankersen, Robust control of large flexible appendages for future space missions, in: Proceedings of 6th International Conference on Dynamics and Control of Systems and Structure in Space, July 2004.

- [33] E. Paolini, M. Battilana, F. Curti, Fast attitude maneuvers and accurate pointing for an ultra-agile spacecraft with flexible appendages actuated by thrusters and reaction wheels, in: *ESA GNC 2011 Conference*, 2011.
- [34] A.A. Shabana, *Dynamics of Multibody Systems*, Cambridge University Press, Cambridge, 2013.
- [35] A. Colagrossi, M. Lavagna, Integrated vibration suppression attitude control for flexible spacecrafats with internal liquid sloshing, *Multibody System Dynamics* 51 (2021) 123–157, <https://doi.org/10.1007/s11044-020-09755-9>.
- [36] C. Angelone, E. Paolini, M. Lavagna, Spacecraft flexible attitude dynamics modelling for accurate control design, in: *ESA GNC 2021 Conference*, 2011.
- [37] F.T. Dodge, *The New Dynamic Behavior of Liquids in Moving Containers*, Southwest Research Institute, San Antonio, 2000.
- [38] R.A. Ibrahim, *Liquid Sloshing Dynamics: Theory and Applications*, Cambridge University Press, Cambridge, 2005.
- [39] H.N. Abramson, W.H. Chu, G.E. Ransleben Jr., Representation of fuel sloshing in cylindrical tanks by an equivalent mechanical model, *ARS Journal* 31 (12) (1961) 1697–1705.
- [40] L. Cederma, E. Paolini, J.D. Biggs, C. Celiberti, Thrusters OFF-Modulation for Attitude Control during Orbital Manoeuvres, *ESA GNC 2021 Conference*, 2011.
- [41] M.P. Christopher, On-orbit performance and operation of the attitude and pointing control subsystems on ASTERIA, in: *Small Satellites Conference (SmallSat 2018)*, Logan, Utah, August 4–9, 2018, 2018.
- [42] NASA Technical Report, *Spacecraft Magnetic Torques*, NASA SP-8018, 1969.
- [43] Z. Zhang, G.S. Aglietti, W. Ren, Coupled microvibration analysis of a reaction wheel assembly including gyroscopic effects in its accelerance, *Journal of Sound and Vibration* 332 (22) (2013) 5748–5765.
- [44] M. Vitelli, B. Specht, F. Boquet, A process to verify the microvibration and pointing stability requirements for the BepiColombo mission, in: *International Workshop on Instrumentation for Planetary Missions* 1683, 2012, p. 1023.
- [45] C. Galeazzi, P.C. Marucchi-Chierro, L.V. Holtz, Experimental activities on ARTEMIS for the microvibration verification, in: *ESA International Conference on Spacecraft Structures, Materials and Mechanical Testing*, Noordwijk, Netherlands, 1996, pp. 997–1006.
- [46] D.-K. Kim, Micro-vibration model and parameter estimation method of a reaction wheel assembly, *Journal of Sound and Vibration* 333 (18) (2014) 4214–4231.
- [47] M.M. Serrano, M. Catania, J. Sánchez, A. Vasconcelos, D. Kuijper, X. Marc, Sentinel-1A flight dynamics LEOP operational experience, in: *International Symposium on Space Flight Dynamics Symposium*, October 2015.
- [48] T. Teil, H. Schaub, Force and Torque Disturbance Modeling Due to General Thruster Plume Impingement, *68th International Astronautical Congress*, Adelaide, Australia, 2017.
- [49] E.C.S.S. Secretariat, *Space Engineering: Control Performance (ECSS-E-ST-60-10C)*, European Cooperation for Space Standardization, 2008. <https://ecss.nl/>.
- [50] ESA, *Pointing Error Engineering Handbook*, 2011. ESSB-HB-E-003.
- [51] K. Yazdi, E. Messerschmid, Analysis of parking orbits and transfer trajectories for mission design of cis-lunar space stations, *Acta Astronautica* 55 (3–9) (2004) 759–771.
- [52] D.A. Vallado, F. David, A critical assessment of satellite drag and atmospheric density modeling, *Acta Astronautica* 95 (2014) 141–165.
- [53] R. Hahn, R. Seiler, Simulating and analyzing the microvibration signature of reaction wheels for future non-intrusive health monitoring methods, in: *Proceedings 14th European Space Mechanisms & Tribology Symposium*, Konstanz, Germany, 2011.

- [54] K.C. Liu, P. Maghami, C. Blaurock, Reaction wheel disturbance modeling, jitter analysis, and validation tests for solar dynamics observatory, in: AIAA Guidance, Navigation and Control Conference and Exhibit, 2008, p. 7232.
- [55] M.P. Le, M.H.M. Ellenbroek, R. Seiler, P. van Put, E.J.E. Cottaar, Disturbances in reaction wheels: from measurements to modeling, in: ESA GNC 2014 Conference, 2014.
- [56] L. Liu, Jitter and basic requirements of the reaction wheel assembly in the attitude control system, *Advances in Space Research* 52 (1) (2013) 222–231, <https://doi.org/10.1016/j.asr.2013.02.014>.

This page intentionally left blank



Orbital dynamics

Andrea Capannolo¹, Stefano Silvestrini¹, Andrea Colagrossi¹,
Vincenzo Pesce²

¹Politecnico di Milano, Milan, Italy

²Airbus D&S Advanced Studies, Toulouse, France

Orbital dynamics describes the translational motion of any object flying in space. Indeed, whenever the spacecraft motion is not directly forced by the control actions computed and actuated by the GNC system, it is affected by the gravitational attractions of the celestial bodies and by the perturbations forces due to the surrounding space environment. The gravity forces are notably larger than the other terms, and, thus, they are primarily characterizing the dynamics in space. According to the Newton's Laws, this results in a spacecraft motion that is always described as an orbit around one or more gravitational attractors. Then, spacecraft natural translational motion is modeled and described exploiting the fundamental rules of the orbital dynamics. The effects of the Guidance, Navigation, and Control (GNC) actions are always directed in modifying the fundamental orbits driving the spacecraft in space. Hence, it is mandatory to master the main concepts of orbital dynamics in order to properly design, implement, and operate a GNC system.

Along its orbit, the spacecraft is theoretically influenced by the gravitational attraction of any existing celestial object. But under a practical perspective, this is obviously not true since the magnitude of many perturbing forces is dramatically larger than the gravity forces exerted by far away planets or other minor celestial bodies. In real world, any spacecraft is primarily orbiting around one main central attractor (i.e., the Earth, Mars, the Moon, the Sun for interplanetary missions, etc.), and its orbit is described according to the two-body problem (2BP) model. This dynamical model can be very simple, modeling the unperturbed Keplerian orbits for preliminary design, or extremely accurate, accounting for any relevant perturbation force.

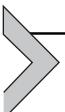
This chapter is divided in four sections:

- *Two-body problem.* This section introduces the main concepts of 2BP dynamics, listing all the useful elements for a GNC engineer. It presents the basic two-body equations, orbit classification, and time laws.

Then, two additional dynamical models are described, since they are becoming very relevant for modern missions and applications: the three-body problem (3BP) and the irregular body dynamics.

- *Three-body problem.* This section introduces the 3BP, which shall be used whenever a mission is set in the region between two comparably relevant gravitational attractors, with the spacecraft orbiting around the so-called Lagrange points. This is becoming very common for missions in Cislunar space, astronomical telescopes in the Sun–Earth system, or exploration missions for other planets’ moons, just to make some examples.
- *Irregular solar system bodies.* This section describes the gravitational environment around irregular solar system bodies. Such modeling technique is mandatory for any space mission around an asteroid, a comet, or another minor solar system object.
- *Relative orbital dynamics.* The chapter is concluded with a section on the relative spacecraft orbital dynamics. In fact, two objects in close orbits can be analyzed describing their relative dynamics under a dedicated point of view. Modern GNC applications are dedicated in solving relative dynamics problems, such as rendezvous, formation flying, or on-orbit servicing.

The reader is invited to deepen the study of spacecraft orbital dynamics in the dedicated books listed in the reference section [1–6].



Two-body problem

The 2BP represents the most known and used model for orbit design and analysis. The model considers two masses only, with the smaller one being of negligible mass with respect to the larger (the main attractor).

The derivation of the 2BP equations of motions starts from the Newton’s law of universal gravitation, applied to a generic distribution of N point masses. The gravitational acceleration acting on the i -th body can be written as [7]:

$$\ddot{\mathbf{r}}_i = - \sum_{j=1, j \neq i}^N \frac{G m_j}{\|\mathbf{r}_{ji}\|^3} \mathbf{r}_{ji}$$

Here, $\ddot{\mathbf{r}}_i$ is the acceleration vector of the i -th mass with respect to the barycenter of the system, G is the gravitational constant, m_j is the mass of the j -th body of the group, and \mathbf{r}_{ji} is the position vector of the i -th body with respect to the j -th body. The above Newton’s inverse square law

predicts that the attraction decreases based on the reciprocal of the square of increasing distance of the i -th body from the j -th point mass, but it eventually does not go to zero. Although we commonly say satellites in orbit experience “zero-gravity,” this is not true. Of course, we think gravity is zero because no normal reaction force is supporting the object. The force of gravity is always present and causes the i -th body (e.g., a satellite) to fall continually around the j -th mass (e.g., the Earth).

In the vast majority of space applications, the motion of the object of interest (an artificial satellite, a moon, etc.) is typically close to a main attractor and rather distant from all other bodies of the cluster. In this context, it is therefore natural to reduce the summation to the mutual attraction of the two close bodies, hence:

$$\begin{aligned}\ddot{\mathbf{r}}_1 &= -\frac{Gm_2}{\|\mathbf{r}_{21}\|^3} \mathbf{r}_{21} \\ \ddot{\mathbf{r}}_2 &= -\frac{Gm_1}{\|\mathbf{r}_{12}\|^3} \mathbf{r}_{12}\end{aligned}\tag{4.1}$$

The previous equation describes the motion of the two isolated masses (or binary system) with respect to their common center of mass; however, for space application, we are interested in how one body moves relatively to the other. This requires an adaptation of the previous dynamics system, exploiting the fact that the barycenter of the binary system is isolated and not subjected to external forces. The motion of the full binary system is described by the following equation:

$$m_1 \ddot{\mathbf{r}}_1 + m_2 \ddot{\mathbf{r}}_2 = 0$$

Considering that $\mathbf{r}_{12} = \mathbf{r}_2 - \mathbf{r}_1$ by definition, it follows that the relative acceleration of body “2” from body “1” reads:

$$\ddot{\mathbf{r}}_2 = \frac{m_1}{m_1 + m_2} \ddot{\mathbf{r}}_{12}\tag{4.2}$$

Substituting Eq. (4.2) to the second equation of the binary system dynamics (Eq. 4.1), the relative motion is obtained, and reads:

$$\ddot{\mathbf{r}}_{12} = -\frac{G(m_1 + m_2)}{\|\mathbf{r}_{12}\|^3} \mathbf{r}_{12}$$

In common applications, the body with mass “ m_2 ” (of which it is desired to define the dynamics) is significantly smaller than the body having the mass “ m_1 ” (the reference attractor). This implies that the overall mass of the binary

system can be approximated with one of the primary gravitational sources, hence:

$$m_1 + m_2 \approx m_1$$

This allows to define the so called “Restricted Two-Body Problem” (R2BP), which reads:

$$\ddot{\mathbf{r}} = -\frac{Gm_1}{\|\mathbf{r}\|^3} \mathbf{r} = -\frac{\mu}{r^3} \mathbf{r} \quad (4.3)$$

where μ is called the “standard gravitational parameter,” $\mathbf{r} = \mathbf{r}_{12}$ and $r = \|\mathbf{r}\|$ for simplicity of notation.

Integrals of motion and orbital elements

The dynamics of Eq. (4.3) system admits some integrals of motion, i.e., relations among the state variables, which are constant over time, that provide isosurfaces where the motion takes place. Such isosurfaces can be exploited to reconstruct the motion in time, without resorting to numerical integration [8]. The search of the integrals of motion allows then to transform the dynamics problem into a set of known parameters as a direct function of the independent variable (i.e., time).

Integrals of motion

Given the three scalar, second-order differential equations of the R2BP, six integration constants have to be defined.

Specific angular momentum

The specific angular momentum can be proven to be an integration constant, by simply verifying that its time derivative is null:

$$\dot{\mathbf{h}} = \frac{d}{dt} (\mathbf{r} \times \dot{\mathbf{r}}) = \mathbf{r} \times \ddot{\mathbf{r}} = \mathbf{r} \times -\frac{\mu}{r^3} \mathbf{r} = 0$$

which is verified for all central forces, such as gravitation (i.e., \mathbf{r} aligned to $\ddot{\mathbf{r}}$).

The condition $\dot{\mathbf{h}} = \mathbf{r} \times \dot{\mathbf{r}} = \mathbf{C}$, with \mathbf{C} being a constant, implies that the motion happens on a fixed plane (i.e., perpendicular to \mathbf{h}).

Eccentricity vector

To define a second integration constant, consider the cross product between $\ddot{\mathbf{r}}$ and \mathbf{h} :

$$\ddot{\mathbf{r}} \times \mathbf{h} = \ddot{\mathbf{r}} \times (\mathbf{r} \times \dot{\mathbf{r}}) = \mu \frac{d}{dt} \left(\frac{\mathbf{r}}{r} \right) \quad (4.4)$$

where the last equality can be verified by substituting the R2BP dynamics to the acceleration term and developing the cross products in the rotating frame.

The left-hand side of Eq. (4.4) can also be rearranged as:

$$\ddot{\mathbf{r}} \times \mathbf{h} = \frac{d\dot{\mathbf{r}}}{dt} \times \mathbf{h} = \frac{d}{dt} (\dot{\mathbf{r}} \times \mathbf{h}) \quad (4.5)$$

Hence, the combination of right-hand sides of Eqs. (4.4) and (4.5) returns:

$$\frac{d}{dt} \left(\dot{\mathbf{r}} \times \mathbf{h} - \frac{\mu}{r} \mathbf{r} \right) = 0$$

The argument of the derivation is therefore constant and represents the expression of the eccentricity vector (or Laplace vector) \mathbf{e} :

$$\mathbf{e} = \frac{1}{\mu} \left(\dot{\mathbf{r}} \times \mathbf{h} \right) - \frac{\mathbf{r}}{r}$$

where the argument has been divided in advance by μ . Notice that, by definition, \mathbf{e} lies on the orbital plane.

Specific energy

The two integrals of motion (\mathbf{h} and \mathbf{e}) provide six conditions, technically enough to fully characterize the motion; however, such quantities are not fully independent, as they are related through the condition $\mathbf{h} \cdot \mathbf{e} = 0$.

Hence, a further scalar condition is needed. This is provided by the conservation of energy. Consider the product of the velocity $\dot{\mathbf{r}}$ and acceleration $\ddot{\mathbf{r}}$, i.e., the specific energy rate of the system:

$$\dot{\mathbf{r}} \cdot \ddot{\mathbf{r}} = \dot{\mathbf{r}} \cdot \left(-\frac{\mu}{r^3} \mathbf{r} \right)$$

The left-hand side is the (specific) kinetic energy rate, and it can be rearranged as $\frac{1}{2} \frac{d}{dt} (\dot{\mathbf{r}} \cdot \dot{\mathbf{r}})$. The right-hand side is the (specific) potential energy rate, and it can be expressed in time derivative terms as $\mu \frac{d}{dt} \left(\frac{1}{r} \right)$. Overall, the specific energy rate reads:

$$\frac{d}{dt} \left(\frac{1}{2} (\dot{\mathbf{r}} \cdot \dot{\mathbf{r}}) - \frac{\mu}{r} \right) = 0$$

Hence, the specific orbital energy is proved to be constant, and its expression reads:

$$\varepsilon = \frac{1}{2} (\dot{\mathbf{r}} \cdot \dot{\mathbf{r}}) - \frac{\mu}{r} \quad (4.6)$$

Orbital elements

From the three integrals of motion (\mathbf{h} , \mathbf{e} , ε), the six-dimensional state of an orbiting object $(\mathbf{r}(t), \dot{\mathbf{r}}(t))$ can be completely described. Nevertheless, one may be interested in finding a more meaningful representation of the motion, through specific orbital parameters (or elements). This can be achieved through the manipulation and elaboration of the integrals of motion. Notice that dimensions of the problem are not affected by the change of parameters; therefore, a set of six parameters is still required.

The constant nature of the specific angular momentum \mathbf{h} implies the existence of a fixed plane on which the orbit lies. Hence, such plane defines a fixed inclination angle “ i ” with respect to the $x - y$ plane of the inertial reference frame in which the state is represented. Furthermore, the intersection of the two planes defines a *Nodal Axis*, \mathbf{N} . The angular distance between the x axis of the inertial frame and \mathbf{N} represents a second fixed parameter Ω , and it is named *Right Ascension of the Ascending Node (RAAN)*. On the orbital plane, \mathbf{N} and \mathbf{e} form a third fixed parameter (the angle between the two vectors), named *Argument of Periapsis*, and labeled ω .

The three parameters i , Ω , and ω uniquely define the *orientation* of the orbit; however, no information about its shape is provided so far.

On the orbital plane, the norm of the specific angular momentum can be expressed in a simpler way leveraging a rotating reference frame with two of its axes aligned with the position r and with h itself $(\hat{\mathbf{r}} - \hat{\theta}\hat{\mathbf{\Theta}} - \hat{\mathbf{h}})$:

$$\begin{aligned} \mathbf{r} &= r\hat{\mathbf{r}} \\ \dot{\mathbf{r}} &= \hat{r}\hat{\mathbf{r}} + r\dot{\theta}\hat{\mathbf{\Theta}} \\ h &= \|\mathbf{r} \times \dot{\mathbf{r}}\| = r^2\dot{\theta} \end{aligned} \quad (4.7)$$

Since h is constant, the third equation represents a confirmation of the *Kepler's second law of planetary motion* [2]. This equivalence will also be useful to express the remaining orbital parameters.

From the eccentricity vector, it is possible to directly extract its magnitude e , which is used as shape-related parameter of the orbit. To better understand its meaning, consider the eccentricity vector \mathbf{e} and the position vector \mathbf{r} , and define θ as the angle between the two vectors. The following relations hold:

$$\mathbf{r} \cdot \mathbf{e} = r e \cos \theta$$

$$\mathbf{r} \cdot \mathbf{e} = \mathbf{r} \cdot \left[\frac{1}{\mu} (\dot{\mathbf{r}} \times \mathbf{h}) - \frac{\mathbf{r}}{r} \right] = \frac{h^2}{\mu} - r$$

Putting together the right-hand terms of the two equations, it is possible to express the orbital radius as a function of θ :

$$r(\theta) = \frac{h^2/\mu}{1 + e \cos \theta} = \frac{p}{1 + e \cos \theta} \quad (4.8)$$

where p is the *semilatus rectum*, and reads:

$$p = \frac{h^2}{\mu} \quad (4.9)$$

Not only does this relation provide the evolution of the distance from the gravity source along the orbit but also it confines the types of motion existing in the R2BP to the *conics* subset (e.g., circles, ellipses, parabolas, and hyperbolas), thus confirming (and extending) the *Kepler's First Law of Planetary Motion* [2].

From Eq. (4.8), other information can be extracted. The parameter θ (i.e., the angle between position and eccentricity vectors) is called *True Anomaly* and is indeed another orbital parameter (function of time) which defines the position of the orbiting object along its trajectory.

Furthermore, it is possible to find the minimum distance from the gravity source by setting $\theta = 0$:

$$r_p = r(0) = \frac{p}{1 + e} \quad (4.10)$$

In case of closed orbits, a maximum distance (i.e., the apocenter) can also be derived, and it is located at $\theta = \pi$:

$$r_a = r(\pi) = \frac{p}{1 - e} \quad (4.11)$$

Then, a last orbital element, i.e., the *semimajor axis* of the conic, is easily derived from minimum and maximum radii as:

$$a = \frac{r_p + r_a}{2} = \frac{p}{1 - e^2}$$

Also, it follows that:

$$\begin{aligned} r_p &= a(1 - e) \\ r_a &= a(1 + e) \\ p &= a(1 - e^2) \end{aligned} \quad (4.12)$$

Indeed, the semimajor axis of the conic represents the last constant parameter, characterizing the size of the orbit. To prove it, the conservation of specific orbital energy is exploited. As the energy value is preserved along the orbit, it can be computed at any point of the conic. For convenience, the pericenter point is considered, as the velocity is fully tangential (i.e., $\dot{r}_p = 0$), and can be easily expressed in terms of the specific angular momentum, namely:

$$v_p = r_p \dot{\theta}(r_p) = \frac{h}{r_p}$$

Then, the energy reads:

$$\epsilon = \frac{1}{2} \frac{h^2}{r_p^2} - \frac{\mu}{r_p}$$

Recalling that $r_p = a(1-e)$ and $h^2 = p\mu = a(1-e^2)\mu$, the new expression of the specific energy is derived:

$$\epsilon = -\frac{\mu}{2a} \quad (4.13)$$

Eq. (4.13) links the energy associated with an orbit and its size through the semimajor axis value and is valid for all conics.

To recap, the full set of orbital elements (and their meaning) are:

- Semimajor axis (size): a .
- Eccentricity (shape): e .
- Inclination (orientation): i .
- RAAN (orientation): Ω .
- Argument of periapsis (orientation): ω .
- True anomaly (location along the orbit): $\theta(t)$.

The R2BP motion can be therefore fully defined through five constant quantities (identifying the orbit) and a sixth parameter (the true anomaly), function of time. Despite the relation between the true anomaly and time can be fully described through the presented integrals of motion, its derivation deserves a dedicated analysis in a separate section.

As a final remark, it is worth mentioning that sometimes the true anomaly is substituted with an alternative parameter, called *mean anomaly*:

$$M = \sqrt{\frac{\mu}{a^3}} \Delta t = n \Delta t$$

with $n = \sqrt{\frac{\mu}{a^3}}$ being the *mean motion* (or *mean angular velocity*). The physical meaning of the mean anomaly will become clear when the time laws will be derived in the next sections.

Two-line elements

Despite the wide utilization of the orbital elements as defined in the previous section, alternative sets have been developed depending on the applications.

For practical use and GNC applications, it is common to find the so-called *Two-Line Elements (TLEs)*. The TLEs resemble the classical elements but display some relevant differences. As the name suggests, parameters are organized in two lines for a standardized automatic processing. They display the alternative set of the six parameters, which are necessary to define the orbit, and they present additional quantities to describe the effect of perturbations on satellites' motion or needed to categorize the data.

In particular, the parameters set found in TLEs is $(n, e, i, \Omega, \omega, M)$. From classical elements, the semimajor axis a has been substituted with the mean motion n (with the Kozai mean value of the semimajor axis [9]), and the true anomaly θ is now replaced by the mean anomaly M .

The additional elements are:

- Satellite number.
- Classification (U: unclassified data).
- International designator (launch year, launch number of the year, piece of the launch).
- Reference epoch of the time-varying elements (year, day of the year, and fractional portion of the day).
- First and second derivatives of the mean motion $\left(\frac{\dot{n}}{2}, \frac{\ddot{n}}{6}\right)$.
- Mean ballistic coefficient $\left(B^* = \frac{1}{2} \frac{c_D A}{m} \rho_0 R\right)$.
- Ephemeris type (orbital model) to generate the data.
- Element number.
- Revolution number at epoch.
- Checksum (used for errors checking).

With the only exception of the satellite number and the checksum (which appear in both lines of the set, respectively, as first and last element), the first line presents the previously listed elements from the classification to the element number (in the same order), while the second line displays the orbital elements first, in the order $\{i, \Omega, e, \omega, M, n\}$, plus the number of revolutions at epoch.

Below is a typical structure of a TLE:
NOAA 14

```
1 23455U 94089A 97320.90946019 .00000140 00000 -0 10191 -3 0.2621
2 23455 99.0090 272.6745 0008546 223.1686 136.8816 14.11711747148495
```

Geometrical classification of the conics

Any conic can be fully defined by two parameters; however, there are various way to express such parameters. As an example, consider the ellipse in Fig. 4.1:

A straightforward approach is to consider the semimajor axis and semiminor axis of the conic, i.e., the couple a and b . Alternatively, one can substitute b (for example) with some parameters linked to the foci distance from the ellipse center, i.e., the focal half-distance c , such that:

$$c^2 = a^2 - b^2$$

The ratio between the focal half-distance and the semimajor axis defines the oblateness of the conic, i.e., the eccentricity e :

$$e = \frac{c}{a}$$

From the previous equations, it follows also an alternative expression of the semiminor axis:

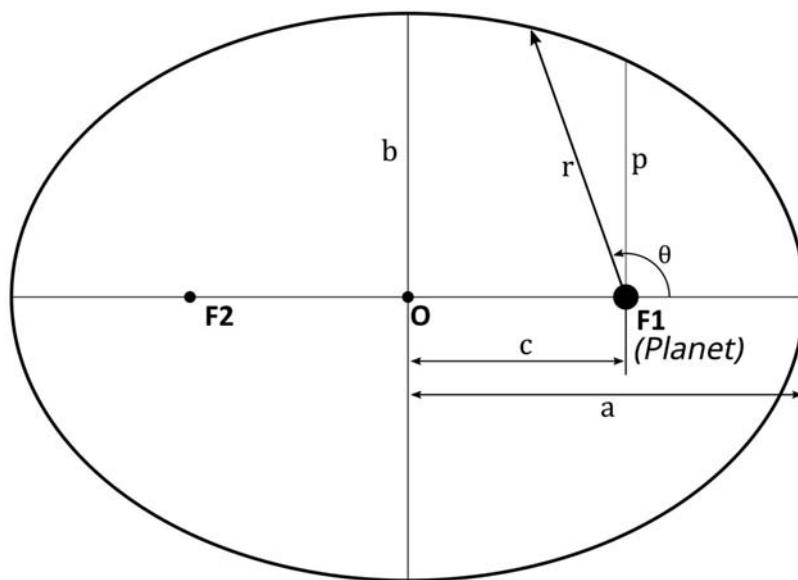


Figure 4.1 Conic's geometrical parameters.

$$b = a\sqrt{1 - e^2}$$

With reference to previous section, a further parameter characterizing a conic is the semilatus rectum, i.e., the distance between the focus and the conic itself in the direction perpendicular to the eccentricity vector. It can be defined by geometrical means (Pythagorean theorem) and leveraging previous quantities as:

$$p = \frac{b^2}{a} = a(1 - e^2)$$

Notice that the same expression was obtained in previous section ([Eq. 4.12](#)), leveraging the orbital radius equation.

Overall, the set of parameters that can be used to characterize a conic are:

- a (semimajor axis)
- b (semiminor axis)
- c (half-distance between foci)
- p (semiparameter or semilatus rectum)
- e (eccentricity)

By leveraging the derived relations from integrals of motion, geometrical parameters can be used to classify the type of conic and, as a consequence, of orbit.

Let's consider the expression of specific energy as a function of semimajor axis ($\epsilon = -\frac{\mu}{2a}$). Then the following conditions are verified:

- $\epsilon < 0 \Leftrightarrow a > 0$
- $\epsilon = 0 \Leftrightarrow a \rightarrow \infty$
- $\epsilon > 0 \Leftrightarrow a < 0$

By extracting a from the energy equation and substituting it to the semilatus rectum expression, the eccentricity e can be defined as:

$$e^2 = 1 + \frac{2pe}{\mu}$$

Recalling also that $p = \frac{h^2}{\mu} > 0$ (strictly positive for orbital motion), the previous expression implies that:

- $\epsilon < 0 \Leftrightarrow 0 \leq e < 1$
- $\epsilon = 0 \Leftrightarrow e = 1$
- $\epsilon > 0 \Leftrightarrow e > 1$

Notice that, since e^2 is by definition a positive quantity, it implies that a minimum energy value exists to ensure orbital motion, when $e = 0$.

Extending the concept to the focal distance, we obtain that:

- $\epsilon < 0 \Leftrightarrow c < a$
- $\epsilon = 0 \Leftrightarrow c = a \rightarrow \infty$
- $\epsilon > 0 \Leftrightarrow c > a$

To understand the conic type from to the geometrical parameters' ranges, we can exploit the orbital radius expression from Eq. (4.8), where the semilatus rectum is replaced by the expression of Eq. (4.12):

$$r(\theta) = \frac{a(1 - e^2)}{1 + e \cos \theta}$$

The minimum energy case implies $e = 0$ and $a > 0$, hence $c = 0$, and the radius reads:

$$r = a$$

This means that the foci are collapsed to the center of the conic, and that the distance from the gravity source is constant and does not depend on the true anomaly; hence, the orbiting object describes a *circular orbit*.

Within the range $0 < e < 1$ instead, the radius expression maintains its original form, and the two foci are distinct, and displaced from the center. Furthermore, the denominator of the expression is always positive, thus admitting a maximum and minimum distance, i.e., a percenter and an apocenter, already defined in Eqs. (4.10) and (4.11). These characteristics identify an *elliptical orbit*.

In the case $e = 1$ and $a \rightarrow \infty$, the focus and the center of the conic collapse at infinity. Here, it is still possible to find a minimum distance r_p as:

$$r_p = \frac{p}{2}$$

Notice that p is undefined if expressed in terms of a and e (Eq. 4.12) and requires the formulation in terms of the angular momentum of the orbit (Eq. 4.9).

On the contrary, the denominator of the radius equation here approaches zero at its minimum; therefore, the maximum distance is at infinity. There, the velocity tends to zero. In fact, $e = 0$ in this case, and if the original expression of the specific energy is evaluated at infinity, we obtain:

$$0 = \frac{1}{2} (\dot{\mathbf{r}} \cdot \dot{\mathbf{r}}) - \frac{\mu}{r} \xrightarrow{r \rightarrow \infty} \frac{1}{2} (\dot{\mathbf{r}} \cdot \dot{\mathbf{r}}) \rightarrow 0$$

Such behavior characterizes the *parabolic orbit*, which is the minimum energy escaping trajectory, and represents a degenerate case of *hyperbolic trajectories*.

Hyperbolic trajectories are described by a semimajor axis $a < 0$ and eccentricity $e > 1$. These escaping trajectories are characterized by a residual velocity at infinite ($v_\infty = \sqrt{2e}$), asymptotically achieved at a specific true anomaly. In particular, the asymptotes of the hyperbola are characterized by a *deflection angle* δ , i.e., the angle measured from the perpendicular axis \hat{p} to the eccentricity vector, and the asymptote. Then, at infinite the true anomaly reads:

$$\vartheta_\infty = \frac{\pi}{2} + \delta$$

Furthermore, such angle can be directly related to eccentricity, by substituting $r \rightarrow \infty$ to the radius equation and extracting θ_∞ , namely:

$$\theta_\infty = \arccos\left(-\frac{1}{e}\right)$$

[Table 4.1](#) summarizes the characteristics of the conics for the R2BP.

Energetic analysis and cosmic velocities

Information about the conics can also be extracted through a purely energetic approach, by leveraging the expression of the specific orbital energy of [Eq. \(4.6\)](#). Furthermore, additional information about orbital velocities can be extracted by analyzing the kinetic component of the energy.

Extracting the radial component of the velocity from the kinetic energy, and leveraging the expression of the specific angular momentum from [Eq. \(4.7\)](#), one can write:

$$\frac{1}{2}\dot{r}^2 = \epsilon - \left(\frac{h^2}{2r^2} - \frac{\mu}{r}\right)$$

Calling $v' = \frac{1}{2}\dot{r}^2$, it is always verified that $v' \geq 0$. Notice that $v' = 0$ implies a local purely tangential velocity of the orbiting body with respect to the gravitational source. [Fig. 4.2](#) depicts the locus of points where such condition is verified, for any distance r .

Hence, for a given angular momentum and planetary constant, the minimum energy allowed at each radius (to have an orbital motion) is provided by the $v' = 0$ line, and reads:

$$\epsilon(r) = \frac{h^2}{2r^2} - \frac{\mu}{r} \quad (4.14)$$

The lowest value of the plot is the minimum of the $\epsilon(r)$ curve. There, a single value of radius can exist; therefore, only a purely circular motion

Table 4.1 Characteristic parameters of the conics.

Orbit	Semimajor axis	Eccentricity	Focal distance	Specific energy	True anomaly range	Radius range
Circle	$a > 0$	$e = 0$	$c < a$	$\varepsilon < 0$	$\theta \in [0, 2\pi]$	$r = a$
Ellipse	$a > 0$	$0 < e < 1$	$c < a$	$\varepsilon < 0$	$\theta \in [0, 2\pi]$	$\frac{p}{1+e} < r < \frac{p}{1-e}$
Parabola	$a \rightarrow \infty$	$e = 1$	$c = a (\rightarrow \infty)$	$\varepsilon = 0$	$\theta \in (-\pi, \pi)$	$\frac{p}{1+e} < r < \infty$
Hyperbola	$a < 0$	$e > 1$	$c > a$	$\varepsilon > 0$	$\theta \in \left(-a \cos\left(-\frac{1}{e}\right), a \cos\left(-\frac{1}{e}\right) \right)$	$\frac{p}{1+e} < r < \infty$

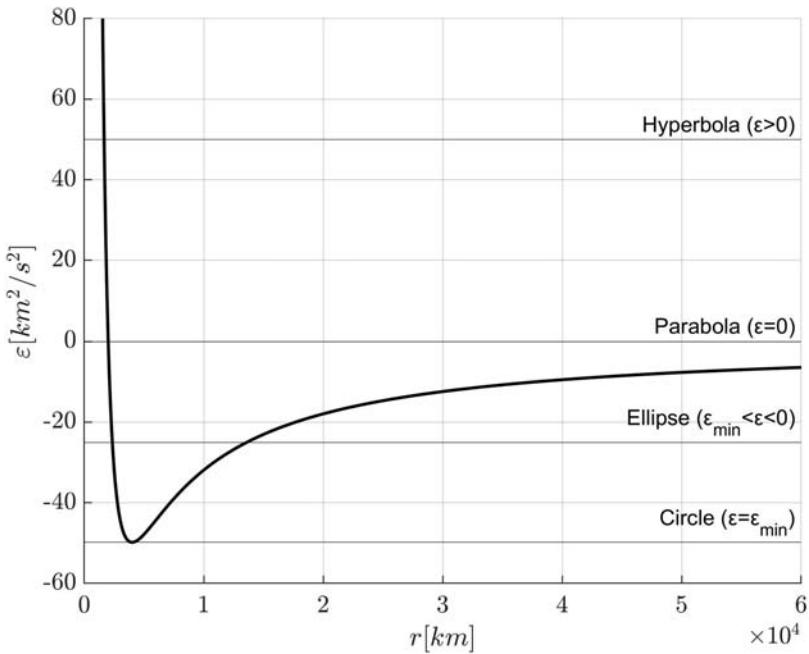


Figure 4.2 Specific orbital energy with null radial velocity, as a function of orbital radius.

around the gravity source is allowed (circular orbit). The orbital velocity is purely tangential (perpendicular to the radius); therefore, its magnitude is directly related to the angular momentum:

$$\nu^2 = \left(\frac{h}{r}\right)^2 \quad (4.15)$$

Recalling Eq. (4.13), and remembering that for circles the semimajor axis coincides with the radius, the energy equation can be solved for ν , and returns:

$$\nu = \sqrt{\frac{\mu}{a}} = \sqrt{\frac{\mu}{r}}$$

which is known as *First Cosmic Velocity*, and often labeled as C_1 .

If the energy level is increased, two distinct intersections with the $\epsilon(r)$ curve are found. This implies that the radial-velocity kinetic energy (ν') can have positive values and equals zero in two distinct points of the graph, i.e., at two distinct radii. Hence, the motion oscillates between a maximum

and a minimum radius where the velocity is purely tangential, which identifies an elliptical orbit.

When the $\epsilon = 0$ level is approached, a single intersection is identified. Indeed, the curve approaches the zero value at infinity; hence, a minimum radius is defined, while the maximum radius tends to infinity. At infinity, the energy equation reads:

$$0 = \frac{1}{2}(\dot{\mathbf{r}} \cdot \dot{\mathbf{r}}) - \frac{\mu}{r} \xrightarrow{r \rightarrow \infty} \frac{1}{2}(\dot{\mathbf{r}} \cdot \dot{\mathbf{r}}) \rightarrow 0$$

This means that, at infinity, the orbital velocity approaches zero. The orbit represents the minimum energy escaping trajectory, which is the parabolic orbit.

If we leverage Eqs. (4.14) and (4.15), the velocity magnitude at pericenter can be extracted, and reads:

$$v = \sqrt{\frac{2\mu}{r_p}}$$

This also known as *Second Cosmic Velocity* and represents the minimum orbital speed required for escaping the gravitational source.

Finally, consider energy levels in the positive region of the graph. Again, a single intersection is identified, representing the minimum orbital radius of the orbit. However, even at $r \rightarrow \infty$, a gap between the energy level and the $\epsilon(r)$ curve is observed. This implies that a residual radial velocity is present at escape conditions; therefore, a hyperbolic orbit is obtained.

The residual escape velocity can be computed from Eqs. (4.8) and (4.6), knowing that the gravitational potential term tends to zero:

$$v = \sqrt{-\frac{\mu}{a}}$$

This velocity is also known as *Third Cosmic Velocity*, and it is widely used in the design of interplanetary transfers. Notice that the value within the square root is still positive, as for hyperbolas, the semimajor axis is negative.

Operative classification of orbits

Geometrical classification of orbits, despite useful to understand the basics of orbital motion, does not provide enough insight when real applications need to be studied. Indeed, orbits can be classified by operative means according to the environment in which they exist, and to their size and orientation. It is worth noticing that because of this correlation with the environment,

different attractors typically imply the existence of different classes of orbits. Here, the main Earth-centered orbits are listed and described.

Low Earth orbits

All circular and quasicircular orbits around the Earth, with an altitude below 2000 km are labeled as low Earth orbits (LEOs). Their close distance to surface and the relatively short orbital period make them a suitable solution for convenient transportation, telecommunication, and Earth surface observation. Main drawbacks are the small field of view from surface (affecting visibility time windows between ground station and satellites) and the fast orbital decay due to the high influence of atmospheric drag and planet's oblateness, which demands frequent orbital control.

Geosynchronous/geostationary orbits

Geosynchronous orbits (GEOs) around the Earth are characterized by a specific and narrow altitude range (around 35,786 km), such that the orbital period matches the Earth rotation. Consequently, a spacecraft on a GEO will maintain nearly the same geographical location with respect to Earth's surface. To exploit this characteristic, GEOs shall have a low or null inclination with respect to Earth equator and a low eccentricity value. A perfectly circular, equatorial orbit at the altitude of geosynchronous family is called geostationary orbit, where a spacecraft would virtually have a fixed zenith with respect to the planet's surface.

Because of synchronicity, and of the large portion of Earth surface visible at this altitude, GEOs are mainly used for telecommunication assets, as ground stations are always in-sight and require minor reorientation of the antennas. In this environment, the Earth oblateness effect and atmospheric drag are negligible, while solar radiation pressure and third-body perturbation from the Moon become more significant.

It is worth mentioning that a match between orbital period and spin of the main attractor can be found for other celestial bodies. For example, Areosynchronous and Areostationary orbits can be found around Mars, about 17,000 km above the Martian surface.

Medium Earth orbits

All the orbits between the altitude ranges of LEO and GEO are classified as medium Earth orbits (MEOs). This region is most commonly used by navigation constellations, such as Galileo, GPS, and GLONASS, as well as by communication satellites.

The MEO region is largely occupied by the Van Allen radiation belts; therefore, additional shielding is typically required from spacecraft populating MEOs.

Sun-synchronous orbits

At high inclination with respect to the Earth equator, and relatively low altitudes, there exists a specific condition where the oblateness of the planet perturbs the orbital plane, making it rotate at the same rate of the Earth orbit around the Sun. This kind of orbit is known as a Sun-synchronous orbit (SSO) and displays a nearly fixed orientation with respect to the Sun direction (and the terminator), thanks to the perturbative effect of the Earth's shape.

This property becomes particularly useful when illumination conditions are important (e.g., for imaging, weather, and reconnaissance satellites) and provide a stable and repeated light-shadow cycle, useful for the design of power and thermal aspects of the spacecraft.

As for GEOs, SSO can be found around other attractors (such as Mars), with properly tuned altitudes and inclinations to accommodate the different oblateness of the planet.

Time laws and orbital period

The orbital motion has been described so far through six parameters derived from the integrals of motion. In particular, five of them are constants that identify the orbit, while the sixth (the true anomaly) is a function of the time and identifies the actual position of the orbiting object along the orbit. To fully define the motion, it is therefore necessary to find the law that links the true anomaly θ to the time t .

Recalling the specific angular momentum and the orbital radius ([Eqs. 4.8 and 4.7](#)), one can define a differential relation between θ and t , namely:

$$h = \left(\frac{p}{1 + e \cos \theta} \right)^2 \dot{\theta} \quad (4.16)$$

Then, the time law is provided by the integration of such relation. The solution of [Eq. \(4.16\)](#) depends on the type of orbit, i.e., on the eccentricity value. The following paragraphs describe the solving approach for the four conics (circles, parabolas, ellipses, and hyperbolas).

Circular orbits

In case of a circular orbit ($e = 0$), the time law is analytical and easily obtained by solving the integral. In fact, the denominator of the right-hand side becomes equal to 1, and the integration reads:

$$\int_0^\theta d\theta = \int_{t_0}^{\Delta t} \frac{h}{p^2} dt$$

where t_0 represents the *time at pericenter*, commonly chosen as reference. Since neither h nor p depends on time, and leveraging the semilatus rectum definition as in Eq. (4.12), the time law for circular orbits is obtained:

$$\theta(t) = \sqrt{\frac{\mu}{a^3}} \Delta t$$

Notice that if the integration over the full orbit is performed, the expression of the orbital period is derived:

$$T = 2\pi \sqrt{\frac{a^3}{\mu}} \quad (4.17)$$

This implies that the period of the orbit depends on its semimajor axis only, and in particular that the squared value of the period is proportional to the cube of the semimajor axis, thus proving the *Kepler's Third Law* [2].

Parabolic orbits

In case of a parabola ($e = 1$), the integrals read:

$$\int_{t_0}^{\Delta t} h dt = \int_0^\theta \left(\frac{p}{1 + \cos \theta} \right)^2 d\theta$$

Here, the solution can be achieved by leveraging trigonometric properties, so that:

$$r(\theta)|_{\text{parabola}} = \frac{p}{1 + \cos \theta} = \frac{1}{2} p \left(1 + \tan^2 \left(\frac{1}{2} \theta \right) \right)$$

Then, the integration returns:

$$\tan^3 \left(\frac{1}{2} \theta \right) + 3 \tan \left(\frac{1}{2} \theta \right) = 3 \sqrt{\frac{\mu}{p^3}} \Delta t \quad (4.18)$$

which is known as *Barker's Equation*.

To solve Eq. (4.18), various methods were developed, such as the *Jerome Cardan's Method*, the *François Viète's Method*, the *Karl Stumpff's Method*, and others [3].

Elliptic orbits

In case of elliptic orbits ($0 < e < 1$), a direct integration does not allow to define a time law. Here, an auxiliary variable needs to be defined. With reference to Fig. 4.3, consider the circle centered at the ellipse's center and having a radius equal to the ellipse's semimajor axis.

We define the *Eccentric Anomaly*, E , as the angle formed by the orbit's periapsis and the position vector on the circle (from center) whose projection on the periapsis coincides to that of the real position along the ellipse. The cartesian components of the ellipse can be expressed as the function of E as:

$$x = a \cos E$$

$$y = b \sin E$$

It is easy to verify that any position (x, y) on the ellipse satisfies the following conditions:

$$a \cos E = ae + r \cos \theta \quad (4.19)$$

$$b \sin E = r \sin \theta \quad (4.20)$$

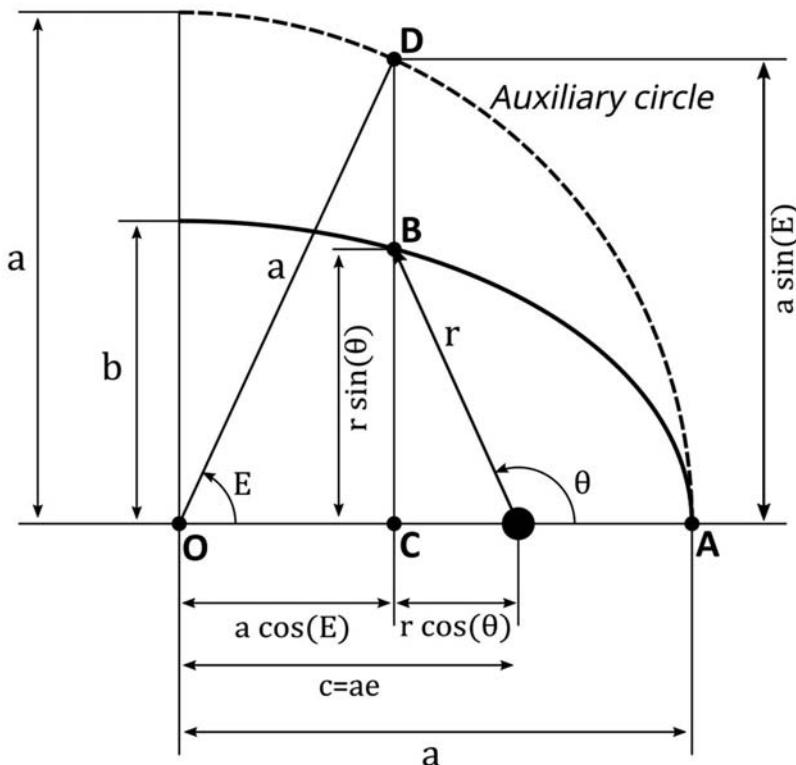


Figure 4.3 Auxiliary circle and variables, built on the reference elliptic orbit (first quadrant depicted).

From the radius Eq. (4.8), the equivalence “ $r \cos \theta = \frac{p-r}{e}$ ” is derived; hence, a new expression of the radius as a function of E can be obtained from Eq. (4.19), and reads:

$$r = a(1 - e \cos E) \quad (4.21)$$

By equating Eqs. (4.8) and (4.21), and leveraging Eqs. (4.19) and (4.20), the following relation between the true anomaly and the eccentric anomaly is obtained:

$$\tan\left(\frac{1}{2}\theta\right) = \sqrt{\frac{1+e}{1-e}} \tan\left(\frac{1}{2}E\right) \quad (4.22)$$

Notice that despite other forms of the relation exist, this is particularly useful as $\frac{1}{2}\theta$ and $\frac{1}{2}E$ are always within the same quadrant.

With a relation available between the two angles, the problem of finding the time law can be expressed in terms of the eccentric anomaly. In particular, if the derivatives in time of the radius expression in Eq. (4.8) and the equivalent expression in Eq. (4.21) are evaluated and compared, it is possible to verify that:

$$\sqrt{\frac{\mu}{p}} \sin \theta = a \dot{E} \sin E \quad (4.23)$$

Exploiting Eqs. (4.20) and (4.21), and eliminating the true anomaly from 23, the following integral equation is derived:

$$\int_{t_0}^t \sqrt{\frac{\mu}{a^3}} dt = \int_0^E (1 - e \cos E) dE$$

and its integration returns:

$$\sqrt{\frac{\mu}{a^3}} \Delta t = E - e \sin E$$

which is known as the *Kepler's Time Law*.

As shown for the circular orbit case, if the integral is computed for the full orbit, the same expression as Eq. (4.17) is obtained, thus confirming the generality of the Kepler's Third Law.

Hyperbolic orbits

In the case of hyperbolic orbits ($e > 1$), a similar approach to the elliptic orbits case is followed. However, here hyperbolic functions are used instead of trigonometric functions, hence:

$$\begin{aligned} x &= a \cosh H \\ y &= b \sinh H \\ r &= a(1 - e \cosh H) \end{aligned} \tag{4.24}$$

with $H = iE$, being i the imaginary number.

The following relation holds between H and the true anomaly θ :

$$\tan \frac{1}{2}\theta = \sqrt{\frac{e+1}{e-1}} \tanh \frac{1}{2}H$$

Finally, by evaluating the time derivative of r in terms of its equivalent expressions (Eqs. 4.8 and 4.24), and eliminating the true anomaly from the equivalence, the integration returns the new time law:

$$\sqrt{\frac{\mu}{-a^3}} \Delta t = e \sinh H - H$$

which is the *hyperbolic form of Kepler's time law*.

Universal time law

The main drawback of the time laws presented in the previous section is the necessity of a different law for each orbital type. It is therefore desirable to have a general expression, applicable regardless of the eccentricity of the orbit. This can be obtained through the so-called *universal parameter* (or *variable*). The change to the universal variable is achieved through the *Sundman's transformation*.

Consider the energy Eqs. (4.6) and (4.13) and recall the angular momentum equivalence $h = \sqrt{\mu p} = r^2\dot{\theta}$. Then, the time derivative of the orbital radius reads:

$$\dot{r}^2 = -\frac{\mu p}{r^2} + \frac{2\mu}{r} - \frac{\mu}{a} \tag{4.25}$$

The objective is to substitute the time derivative, with the universal variable derivative of r . Define the universal variable χ such that:

$$\dot{\chi} = \frac{\sqrt{\mu}}{r}$$

$$\chi(0) = 0$$

Then, Eq. (4.25) becomes:

$$\left(\frac{dr}{d\chi}\right)^2 = -p + 2r - \frac{r^2}{a}$$

whose integral returns $r(\chi)$:

$$r(\chi) = a \left(1 + e \sin \left(\frac{\chi + C_0}{\sqrt{a}} \right) \right) \quad (4.26)$$

This result is valid only for positive semimajor axis a . Also, the eccentricity e is still present, and the integration constant C_0 is to be determined. The next steps allow to completely remove the dependency on the conic type and to generalize the previous expression.

First, substitute the expression in Eq. (4.26) into the time derivative of χ to obtain the relationship between the universal variable and time:

$$\sqrt{\mu} \Delta t = a \chi - ae \sqrt{a} \left\{ \cos \left(\frac{\chi + C_0}{\sqrt{a}} \right) - \cos \left(\frac{C_0}{\sqrt{a}} \right) \right\} \quad (4.27)$$

Then, if $r(\chi)$ and $\dot{r}(\chi)$ are evaluated for the initial condition $t = 0$, recalling that $\chi(0) = 0$, the following relations hold:

$$e \sin \left(\frac{C_0}{\sqrt{a}} \right) = \frac{r_0}{a} - 1$$

$$e \cos \left(\frac{C_0}{\sqrt{a}} \right) = \frac{\mathbf{r}_0 \cdot \dot{\mathbf{r}}_0}{\sqrt{\mu a}}$$

Substituting these expressions in Eqs. (4.26) and (4.27), the final relation between r and χ , and the *universal time law* are defined:

$$r(\chi) = a \left\{ 1 + \frac{\mathbf{r}_0 \cdot \dot{\mathbf{r}}_0}{\sqrt{\mu a}} \sin \left(\frac{\chi}{a} \right) + \left(\frac{r_0}{a} - 1 \right) \cos \left(\frac{\chi}{\sqrt{a}} \right) \right\}$$

$$\sqrt{\mu} \Delta t = a \chi - a \sqrt{a} \left\{ \frac{\mathbf{r}_0 \cdot \dot{\mathbf{r}}_0}{\sqrt{\mu a}} \cos \left(\frac{\chi}{\sqrt{a}} \right) - \left(\frac{r_0}{a} - 1 \right) \sin \left(\frac{\chi}{\sqrt{a}} \right) - \frac{\mathbf{r}_0 \cdot \dot{\mathbf{r}}_0}{\sqrt{\mu a}} \right\}$$

Summary

Table 4.2 collects the final expressions for the time law, derived in the previous paragraph.

Table 4.2 Time law solution methods.

Time law	Solving equations
Circular orbits	$\theta(t) = \sqrt{\frac{\mu}{a^3}}\Delta t$
Parabolic orbits	$\tan^3\left(\frac{1}{2}\theta\right) + 3 \tan\left(\frac{1}{2}\theta\right) = 3\sqrt{\frac{\mu}{P^3}}\Delta t$
Elliptic orbits	$\tan\left(\frac{1}{2}\theta\right) = \sqrt{\frac{1+e}{1-e}}\tan\left(\frac{1}{2}E\right)$ $\sqrt{\frac{\mu}{a^3}}\Delta t = E - e \sin E$
Hyperbolic orbits	$\tan\frac{1}{2}\theta = \sqrt{\frac{e+1}{e-1}}\tanh\frac{1}{2}H$ $\sqrt{\frac{\mu}{-a^3}}\Delta t = e \sinh H - H$
Universal variable	$r(\chi) = a\left\{1 + \frac{\mathbf{r}_0 \cdot \dot{\mathbf{r}}_0}{\sqrt{\mu a}} \sin\left(\frac{\chi}{a}\right) + \left(\frac{\mathbf{r}_0}{a} - 1\right) \cos\left(\frac{\chi}{\sqrt{a}}\right)\right\}$ $a\sqrt{a}\left\{\frac{\mathbf{r}_0 \cdot \dot{\mathbf{r}}_0}{\sqrt{\mu a}} \cos\left(\frac{\chi}{\sqrt{a}}\right) - \left(\frac{\mathbf{r}_0}{a} - 1\right) \sin\left(\frac{\chi}{\sqrt{a}}\right) - \frac{\mathbf{r}_0 \cdot \dot{\mathbf{r}}_0}{\sqrt{\mu a}}\right\}$
$\sqrt{\mu}\Delta t = a\chi -$	

Orbital perturbations

The previous section dealt with the full characterization of the R2BP, in terms of dynamics equations, integrals of motion, and parametric representation of the solutions (conics) of the problem. Furthermore, insight has been given about energy-related features of orbits, and time laws defined.

All these results have been derived in the context of an unperturbed environment, where only single attractor gravitational force is present. In practice, orbits are subjected to a series of perturbation, which deviate the nominal motion and that must be considered for higher fidelity simulations. The most relevant perturbations and their modeling rules have been described in [Chapter 3 – The Space Environment](#). Here, the way in which they affect the orbital dynamics and the methods to compute their effects are presented.

Indeed, the effects of perturbation can be taken into account following two main approaches (although hybrid techniques are also available), namely through an analytical formulation or a numerical one. In the next subsections, the main analytical and numerical approaches are formulated and described.

A numerical approach: the Cowell's formulation

The numerical formulation is of simpler derivation, as it directly acts on the equations of motion, through the addition of the perturbing term.

Furthermore, it generally provides accurate results, regardless of the orders of magnitude of the perturbative effects. On the other hand, its formulation implies dependency on the initial conditions (hence, a loss of generality of the dynamical behavior), and a generally higher effort from the computational point of view.

The Cowell's formulation represents the easiest approach and consists of a simple addition of the perturbative accelerations \mathbf{f} to the R2BP equations of motion (Eq. 4.3):

$$\ddot{\mathbf{r}} = -\frac{\mu}{r^3} \mathbf{r} + \mathbf{f}$$

Hence, the complexity of this approach (if there is any) is to choose a proper numerical propagation method, which suits the desired accuracy and speed of the process.

Noteworthy suitable methods for orbital propagation are the predictor–corrector schemes, and in particular the Verner's Runge–Kutta eighth(seventh) order method [10], and the variable-step/variable-order Adam–Bashforth–Moulton solver of orders from 1st to 13th [11].

An analytical approach: Gaussian Variation of Parameters

The analytical formulation relies on the fact that perturbative forces are significantly smaller than the main gravitational force; hence, they can be locally approximated by series expansions. This implies that the orbital motion can still be locally expressed in terms of the orbital parameters (*osculating elements*), but that such parameters are no longer constant and that they possess some rate of change with time, that is:

$$\frac{d\mathbf{c}}{dt} = f(\mathbf{c}, t)$$

where \mathbf{c} represents the vector collecting the six orbital elements $(a, e, i, \Omega, \omega, \theta)$.

The mathematical derivation provides general formulas which are valid regardless of the initial conditions. The more complex development is compensated by the insight given by the formulas about the local behavior of the orbit.

Among the analytical techniques, the Gaussian Variation of Parameters (VOPs) provides relatively simple expressions, applicable in the presence of both conservative and nonconservative forces (as opposed, e.g., to the Lagrangian VOP [2]). Nevertheless, they have a limited applicability range, as they are suited for closed orbits only (eccentricity less than one) and

present some singularities (related to the orbital angles). Alternative formulations have been developed to deal with singularities, such as the expression of the VOP in terms of *equinoctial elements* [3].

In the following paragraphs, the expression of each parameter's variation is derived as a result of a generic perturbing acceleration “ \mathbf{f} .“ The equations are expressed in a reference frame $(\hat{\mathbf{r}}, \hat{\theta}, \hat{\mathbf{h}})$, with $\hat{\mathbf{r}}$ aligned with orbital position vector and $\hat{\mathbf{h}}$ aligned with the angular momentum vector. For simplicity, the components of the perturbative acceleration along the three axes are named “ f_r, f_θ, f_h .“

Semimajor axis

The rate of variation of the semimajor axis can be derived from the specific energy equation. In particular, the specific energy is not constant anymore due to the presence of the perturbations. Its rate is provided by the product of the perturbative acceleration and the local orbital velocity:

$$\frac{d\varepsilon}{dt} = \mathbf{f} \cdot \dot{\mathbf{r}} = \dot{r} f_r + r \dot{\theta} f_\theta \quad (4.28)$$

Recalling that $\varepsilon = -\frac{\mu}{2a}$, the derivative of the semimajor axis with respect to energy reads:

$$\frac{da}{d\varepsilon} = \frac{\mu}{2\varepsilon^2} = \frac{2a^2}{\mu} \quad (4.29)$$

Leveraging the chain rule of derivatives in Eqs. (4.28) and (4.29), and recalling that

$$\begin{aligned} \dot{r} &= \frac{p}{(1 + e \cos \theta)^2} e \dot{\theta} \sin \theta \\ p &= a(1 - e^2) = \frac{h^2}{\mu} \end{aligned} \quad (4.30)$$

$$h = r^2 \dot{\theta}$$

the time rate of a is provided:

$$\frac{da}{dt} = \frac{2e \sin \theta}{n \sqrt{1 - e^2}} f_r + \frac{2a \sqrt{1 - e^2}}{nr} f_\theta \quad (4.31)$$

$$n = \sqrt{\frac{\mu}{a^3}} \quad (4.32)$$

Eccentricity

To derive the rate of change of the orbital eccentricity, Eq. (4.30) can be used as starting point. From its derivation, and isolating the eccentricity term, the expression reads:

$$\frac{de}{dt} = -\frac{h}{\mu ae} \frac{dh}{dt} + \frac{h^2}{2\mu a^2 e} \frac{da}{dt} \quad (4.33)$$

The first term contains the variation of angular momentum's magnitude, caused by the perturbations. Its expression can be retrieved deriving the \mathbf{h} vector in time:

$$\frac{d\mathbf{h}}{dt} = \dot{h} \hat{\mathbf{h}} + h \dot{\theta} \hat{\boldsymbol{\theta}} \quad (4.34)$$

We are interested in the \dot{h} term. To express it in terms of perturbing acceleration, differentiate again the \mathbf{h} vector, expressed in terms of \mathbf{r} and $\dot{\mathbf{r}}$:

$$\frac{d\mathbf{h}}{dt} = \frac{d}{dt} (\mathbf{r} \times \dot{\mathbf{r}}) = \mathbf{r} \times \ddot{\mathbf{r}} = r f_\theta \hat{\mathbf{h}} - r f_h \hat{\boldsymbol{\theta}} \quad (4.35)$$

Notice that, as expected, the gravitational term cancels out as it is aligned with the position vector \mathbf{r} , proving again that the variation of the angular momentum is preserved in the absence of perturbations.

To extract the expression of $\dot{h} = \frac{dh}{dt}$, the terms along $\hat{\mathbf{h}}$ of Eqs. (4.31) and (4.35) are compared:

$$\frac{dh}{dt} = r f_\theta \quad (4.36)$$

Regarding the second term of Eq. (4.33) is the semimajor axis rate, already defined in Eq. (4.34). Substituting Eqs. (4.34) and (4.36) in Eq. (4.33), and leveraging again the relations in Eqs. (4.30) and (4.32), the final expression of the eccentricity rate is obtained:

$$\frac{de}{dt} = \frac{\sqrt{1-e^2} \sin \theta}{na} f_r + \frac{\sqrt{1-e^2}}{na^2 e} \left[\frac{a^2(1-e^2)}{r} - r \right] f_\theta \quad (4.37)$$

Inclination

To compute the rate of change of the inclination $\frac{di}{dt}$, recall the inertial reference frame $(\hat{\mathbf{I}}, \hat{\mathbf{J}}, \hat{\mathbf{K}})$, such that:

$$\hat{\mathbf{h}} \cdot \hat{\mathbf{K}} = \cos i \quad (4.38)$$

$$\hat{\mathbf{r}} \cdot (\hat{\mathbf{K}} \times \hat{\mathbf{h}}) = \cos(\omega + \theta)$$

Also, leveraging the spherical angles trigonometry, it is possible to derive a relation with respect to $\hat{\theta}$:

$$\hat{\theta} \cdot \hat{\mathbf{K}} = \sin i \cos(\omega + \theta) \quad (4.39)$$

[Eq. \(4.38\)](#) can be modified to include the magnitude of the angular momentum as:

$$\cos i = \frac{\hat{\mathbf{h}} \cdot \hat{\mathbf{K}}}{h} \quad (4.40)$$

The differentiation in time of [Eq. \(4.40\)](#) (here omitted for simplicity) presents the derivative terms $\frac{di}{dt}$, $\frac{dh}{dt}$, and $\frac{dh}{dt} = \dot{h}$, with the first being the new quantity to be determined. The other two quantities have already been defined in [Eqs. \(4.35\)](#) and [\(4.36\)](#). Substituting such quantities, and leveraging the relations in [Eqs. \(4.38\)](#) and [\(4.39\)](#), the final form of the inclination rate is obtained:

$$\frac{di}{dt} = \frac{r \cos(\omega + \theta) f_h}{na^2 \sqrt{1 - e^2}} \quad (4.41)$$

Right ascension of the ascending node

Following the same approach used for the inclination, the RAAN (Ω) can be extracted from vectorial relationships between the two reference frames $(\hat{\mathbf{r}}, \hat{\theta}, \hat{\mathbf{h}})$ and $(\hat{\mathbf{I}}, \hat{\mathbf{J}}, \hat{\mathbf{K}})$:

$$\cos \Omega = \frac{\hat{\mathbf{I}} \cdot (\hat{\mathbf{K}} \times \hat{\mathbf{h}})}{\|\hat{\mathbf{K}} \times \hat{\mathbf{h}}\|} \quad (4.42)$$

$$\hat{\mathbf{I}} \cdot (\hat{\mathbf{K}} \times \hat{\mathbf{h}}) = \cos \Omega \sin i \quad (4.43)$$

$$\hat{\mathbf{I}} \cdot (\hat{\mathbf{K}} \times \hat{\theta}) = \sin \Omega \sin(\omega + \theta) - \cos \Omega \cos(\omega + \theta) \cos i \quad (4.44)$$

Differentiating [Eq. \(4.42\)](#) and substituting the relations in [Eqs. \(4.43\)](#) and [\(4.44\)](#), the expression of $\frac{d\Omega}{dt}$ appears in the terms of $\frac{dh}{dt}$ and $\frac{di}{dt}$. Then, substituting the two quantities with [Eqs. \(4.36\)](#) and [\(4.41\)](#), the final equation for the RAAN variation in time is obtained:

$$\frac{d\Omega}{dt} = \frac{r \sin(\omega + \theta)}{h \sin i} f_h \quad$$

True anomaly

The variation of true anomaly $\frac{d\theta}{dt}$ can be derived from the differentiation of the orbital radius equation:

$$r \left(\frac{de}{dt} \cos \theta - e \sin \theta \frac{d\theta}{dt} \right) = \frac{2h}{\mu} \frac{dh}{dt} \quad (4.45)$$

Notice that the radius r was not differentiated in time, as here we are deriving the local variation due to perturbation and not due to orbital motion. For the same reason, $\frac{d\theta}{dt}$ represents the true anomaly variation as a pure perturbative effect, and it is different from the previously defined $\dot{\theta}$ (related to orbital motion). This approximated and local “separation of effects” further highlights the bounds of applicability of the analytical approach, which quickly loses accuracy as time moves forward.

[Eq. \(4.45\)](#) presents the time derivative of both eccentricity and angular momentum magnitude; hence, by substituting them with [Eqs. \(4.36\)](#) and [\(4.37\)](#), the final time variation of the true anomaly is obtained:

$$\frac{d\theta}{dt} = \frac{\sqrt{1-e^2}}{nae} \cos \theta f_r - \frac{\sqrt{1-e^2}}{nae} \frac{2+e \cos \theta}{1+e \cos \theta} f_\theta. \quad (4.46)$$

Argument of periapsis

Following the same approach as the inclination and the RAAN, the time variation of the argument of periapsis (due to perturbation forces) can be derived from vectorial relationships. Consider the argument of latitude ($\omega + \theta$), formed by the vector $\hat{\mathbf{r}}$ of the rotating frame and by the nodal axis $\hat{\mathbf{N}} = \hat{\mathbf{K}} \times \hat{\mathbf{h}}$. The following relation holds:

$$\cos(\omega + \theta) = \hat{\mathbf{r}} \cdot (\hat{\mathbf{K}} \times \hat{\mathbf{h}}) = \frac{\mathbf{r} \cdot (\hat{\mathbf{K}} \times \mathbf{h})}{r \| \hat{\mathbf{K}} \times \mathbf{h} \|} \quad (4.47)$$

Differentiating [Eq. \(4.47\)](#) will highlight time derivatives of both h and θ , which can be replaced by [Eqs. \(4.36\)](#) and [\(4.46\)](#). The explicit expression of $\frac{d\omega}{dt}$ reads:

$$\frac{d\omega}{dt} = \frac{\sqrt{1-e^2}}{nae} \left[-\cos \theta f_r + \frac{2+e \cos \theta}{1+e \cos \theta} \sin \theta f_\theta \right] - \frac{r \cot i \sin u}{h} f_h$$

Validity range of the two-body problem

The 2BP is valid whenever its fundamental assumption is satisfied: the space-craft motion is influenced by the gravity attraction of one central body and all the other forces can be treated as perturbations. This assumption shall be verified by comparing the gravitational forces of other celestial objects, since all the other nongravitational perturbations are typically few orders of magnitude smaller than the gravity ones.

To verify this, the concept of the sphere of influence (SOI) shall be introduced. The SOI of a celestial body with mass, m , is described as the region of space within which the motion of a spacecraft is influenced by that object's gravity more than any other celestial body with mass, M . For example, the Earth's SOI is found by identifying the region of space where the Earth's gravity is more dominant than the Sun or any other planet's gravity.

The SOI is mathematically defined with its radius:

$$r_{\text{SOI}} = a \left(\frac{m}{M} \right)^{2/5},$$

where a is the semimajor axis of the object with mass m (e.g., a planet, a moon) orbiting around the body with mass M (i.e., the Sun, the main planet). For example, the SOI of the Earth with respect to the Sun is about 924 645 km, and for the Moon with respect to the Earth, it is about 66 183 km.

It is wrong to apply the R2BP outside the SOI of the central planet. When dealing with interplanetary missions and the spacecraft is moving between the neighborhoods of different celestial objects using a two-body approximation (i.e., ellipses and hyperbolae), the SOI is taken as the boundary where the trajectory switches which mass field it is governing the R2BP orbital dynamics. This method is denoted as patched conic trajectory analysis [2]. As an illustrative example, we can consider a patched conic analysis of a trajectory from the Earth to Phobos. At first, the patched conic method would model the spacecraft exiting the Earth's SOI to enter the Sun's SOI. Then, the spacecraft would enter the SOI of Mars, and, finally, while inside the Mars' SOI, it would enter the Phobos' SOI. The motion of the spacecraft within the Phobos' SOI would be modeled using Phobos-centered orbits, treating as perturbations the influence of Mars, as well as the Sun. In fact, the moons of a planet are typically well inside of the planet's SOI, but they have their own SOIs. Inside them, the motion of a spacecraft is governed more by the moon's gravity than by the Sun's or by the planet's gravity.



Three-body problem

The 3BP represents the natural extension of the single attractor dynamics. It is employed in all scenarios where an additional gravitational source is as relevant as the primary one, and it cannot be modeled as a perturbation source (as in [Chapter 3 – The Space Environment, Section Perturbation Sources](#)). With reference to the validity range of the R2BP, this condition is likely to occur at the borders of the spheres of influence, and in any space regions where a central attractor cannot be uniquely identified. 3BP is getting an increased attention for GNC applications, since modern space missions effectively exploit the dynamics offered by 3BP environments, such as the Cislunar space, or the Sun–Earth SOI borders.

Following the same procedure for the 2BP, the general 3BP dynamics can be derived from the expression of the N-body, point mass gravitational acceleration [7], by limiting the number of masses to three. The acceleration acting on the i -th body of the triplet reads:

$$\ddot{\mathbf{r}}_i = - \sum_{j=1, j \neq i}^3 \frac{Gm_j}{\|\mathbf{r}_{ji}\|^3} \mathbf{r}_{ji} \quad (4.48)$$

The full motion of the system is provided by integrating in parallel [Eq. \(4.48\)](#) for $i = 1, 2, 3$.

This expression, although elegant, is characterized by a complex and chaotic behavior when integrated in time and fails to provide a sufficiently regular environment where periodic or quasiperiodic motions (of particular interest for engineering purposes) can be found [12]. To overcome this issue, simplified models are usually exploited.

From an applicative, engineering point of view, we are interested in the motion of only one of the three masses (the spacecraft) as a consequence of the other bodies' gravitation (planets, moons, the Sun, etc.). It is reasonable to assume a negligible mass of the spacecraft than one of the celestial bodies exerting their gravitational force. With reference to [Eq. \(4.48\)](#), assuming that the two massive bodies correspond to $i = 1, 2$, and the spacecraft to $i = 3$, the new dynamical system reads:

$$\begin{aligned} \ddot{\mathbf{r}}_3 &= -\frac{Gm_1}{\|\mathbf{r}_{13}\|^3} \mathbf{r}_{13} - \frac{Gm_2}{\|\mathbf{r}_{23}\|^3} \mathbf{r}_{23} \\ \ddot{\mathbf{r}}_1 &= -\frac{Gm_2}{\|\mathbf{r}_{21}\|^3} \mathbf{r}_{21} \\ \ddot{\mathbf{r}}_2 &= -\frac{Gm_1}{\|\mathbf{r}_{12}\|^3} \mathbf{r}_{12} \end{aligned} \quad (4.49)$$

The new expression is known as “Restricted Three-Body Problem” (RTBP), and it represents the basis of the dynamics models used to preliminary design orbits in binary systems. Within the RTBP, the two attractors are subjected to the 2BP dynamics, and their motion can be decoupled from the third body.

Depending on the attractors’ motion, we talk about “Circular Restricted Three-Body Problem” (CRTBP) or “Elliptic Restricted Three-Body Problem” (ERTBP).

Circular Restricted Three-Body Problem

In the CRTBP, the two attractors’ motion describes circular orbits around the barycenter of the binary system. In such framework, it is convenient to express the equations of motion of the third body in the so called “synodic reference frame” [13], which rotates along with the attractors (or primaries) at the same angular rate. Within such frame, the primaries are fixed, and their dependence on time is eliminated.

By applying the classical rotation rules to Eq. (4.3), the new equations read:

$$\ddot{\mathbf{r}} = \omega^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{r} - 2\omega \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\mathbf{r}} - G \left(\frac{m_1}{\|\mathbf{r} - \mathbf{r}_1\|^3} (\mathbf{r} - \mathbf{r}_1) + \frac{m_2}{\|\mathbf{r} - \mathbf{r}_2\|^3} (\mathbf{r} - \mathbf{r}_2) \right) \quad (4.50)$$

where \mathbf{r} refers to the third body position vector in the synodic reference frame, \mathbf{r}_1 and \mathbf{r}_2 are the primaries position from the barycenter, and ω is the angular velocity of the binary system, computed according to the 2BP as:

$$\omega = \sqrt{\frac{G(m_1 + m_2)}{L^3}} \quad (4.51)$$

with L being the (constant) distance between the two primaries.

It is a common practice to express Eq. (4.50) in nondimensional form, leveraging the constant parameters that characterize the CRTBP. In particular, it is possible to normalize the quantities according to the total mass of the attractors ($m_1 + m_2$), to their distance (L), and to the angular velocity (ω). From Eq. (4.51), it can be observed that such normalization implies setting $G = 1$.

The resulting nondimensional form of Eq. (4.50) reads:

$$\ddot{\hat{\mathbf{r}}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \hat{\mathbf{r}} - 2 \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\hat{\mathbf{r}}} - \frac{1-\mu}{\|\hat{\mathbf{r}}_1\|^3} \hat{\mathbf{r}}_1 + \frac{\mu}{\|\hat{\mathbf{r}}_2\|^3} \hat{\mathbf{r}}_2$$

$$\hat{\mathbf{r}}_1 = \hat{\mathbf{r}} + \begin{bmatrix} \mu \\ 0 \\ 0 \end{bmatrix} \quad (4.52)$$

$$\hat{\mathbf{r}}_2 = \hat{\mathbf{r}} + \begin{bmatrix} \mu-1 \\ 0 \\ 0 \end{bmatrix}$$

with μ being the mass ratio of the primaries, expressed as follows:

$$\mu = \frac{m_2}{m_1 + m_2} \quad (4.53)$$

An even more compact form can be expressed by identifying a “pseudo-potential” function U such that:

$$U = \frac{1}{2} (x^2 + y^2) + \frac{1-\mu}{\|\tilde{\mathbf{r}}_1\|} + \frac{\mu}{\|\tilde{\mathbf{r}}_2\|} \quad (4.54)$$

where x and y represent the planar components of the nondimensional position vector of the third body.

Then, the first expression of Eq. (4.52) reads:

$$\ddot{\hat{\mathbf{r}}} + 2 \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\hat{\mathbf{r}}} = \nabla U \quad (4.55)$$

From Eq. (4.55), it is possible to identify five fixed equilibrium points in the synodic reference frame, also known as “Lagrangian Points.” The equilibrium condition requires that:

$$\nabla U = 0 \quad (4.56)$$

Two of the three conditions of Eq. (4.56) allow to locate the equilibrium points on the binary system's plane ($z = 0$) and identify one of the two coordinates on the plane (typically the y component). Furthermore, three of them result to be aligned with the primaries ($y = 0$), and for this reason, they are referred to as "Collinear Lagrangian Points" (L_1 , L_2 , and L_3).

The third condition is a quintic equation that allows to find also the x component of the equilibria position vectors. In particular, it is observed that the two points not aligned with the primaries (L_4 and L_5) are located at the vertices of two equilateral triangles, with their bases connecting the primaries; hence, their coordinates can be expressed in the synodic frame as a function of the mass ratio:

$$\begin{aligned}\mathbf{r}_{L4} &= \left[-\mu + \frac{1}{2}, \frac{\sqrt{3}}{2}, 0 \right]^T \\ \mathbf{r}_{L5} &= \left[-\mu + \frac{1}{2}, -\frac{\sqrt{3}}{2}, 0 \right]^T\end{aligned}\tag{4.57}$$

For this reason, they are also called "Equilateral Lagrangian Points."

The collinear Lagrangian points are saddle points; hence, they are unstable. The equilateral Lagrangian points possess stable properties if the mass ratio (μ) is below the value 0.03852 [13].

Elliptic Restricted Three-Body Problem

In the ERTBP, the two attractors' motion describes elliptic orbits around the barycenter of the binary system. Similar to the CRTBP, it is possible to define a rotating reference frame which rotates at the same rate of the binary system. However, the elliptic motion of the primaries makes the problem nonautonomous, as their mutual distance and the corresponding rotation rate depend on time.

The time-dependent, dimensional equations of motion of the ERTBP read:

$$\begin{aligned}\ddot{\mathbf{r}} &= \left(\omega(t)^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \dot{\omega}(t) \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \mathbf{r} \\ &\quad - 2\omega(t) \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\mathbf{r}} \\ &\quad - G \left(\frac{m_1}{\|\mathbf{r} - \mathbf{r}_1\|^3} (\mathbf{r} - \mathbf{r}_1) + \frac{m_2}{\|\mathbf{r} - \mathbf{r}_2\|^3} (\mathbf{r} - \mathbf{r}_2) \right)\end{aligned}\tag{4.58}$$

Here, the time-dependent angular velocity is expressed as:

$$\omega(t) = \sqrt{\frac{G(m_1 + m_2)}{L(t)^3}} \quad (4.59)$$

The variable primaries' distance is expressed through the 2BP orbital parameters:

$$L(t) = a(1 - e \cos E(t)) \quad (4.60)$$

with a , e , and E being the semimajor axis, the eccentricity, and the eccentric anomaly of the 2BP orbit.

The presence of the eccentric anomaly in Eq. (4.60) implies that Eq. (4.58) shall be augmented with the variation of such parameter, namely:

$$\dot{E} = \frac{1}{1 - e \cos E(t)} \quad (4.61)$$

A nondimensional form can be expressed also for the ERTBP. Here, the normalization is performed according to the total mass of the primaries ($m_1 + m_2$), to the gravitational constant (G), and to the semimajor axis of the 2BP (a). Consequently, the nondimensional angular velocity, and its derivative, take the form:

$$\begin{aligned} \hat{\omega}(t) &= \frac{\sqrt{1 - e^2}}{(1 - e \cos E(t))^2} \\ \dot{\hat{\omega}}(t) &= -\frac{2e \sqrt{1 - e^2}}{(1 - e \cos E(t))^2} \sin E(t) \end{aligned} \quad (4.62)$$

The nondimensional equations of dynamics read:

$$\begin{aligned} \ddot{\hat{\mathbf{r}}} &= \left(\hat{\omega}(t)^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \dot{\hat{\omega}}(t) \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \hat{\mathbf{r}} - 2\hat{\omega}(t) \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \hat{\mathbf{r}} - \frac{1 - \mu}{\|\hat{\mathbf{r}}_1\|^3} \hat{\mathbf{r}}_1 + \frac{\mu}{\|\hat{\mathbf{r}}_2\|^3} \hat{\mathbf{r}}_2 \\ \dot{E} &= \frac{1}{1 - e \cos E(t)} \\ \hat{\mathbf{r}}_1 &= \hat{\mathbf{r}} + \begin{bmatrix} \mu(1 - e \cos E(t)) \\ 0 \\ 0 \end{bmatrix} \\ \hat{\mathbf{r}}_2 &= \hat{\mathbf{r}} + \begin{bmatrix} (\mu - 1)(1 - e \cos E(t)) \\ 0 \\ 0 \end{bmatrix} \end{aligned} \quad (4.63)$$

Again, Eq. (4.62) can be expressed in a more compact form as:

$$\ddot{\hat{\mathbf{r}}} + \dot{\hat{\omega}}(t) \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \hat{\mathbf{r}} + 2\hat{\omega}(t) \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\hat{\mathbf{r}}} = \nabla U$$

$$\dot{E} = \frac{1}{1 - e \cos E(t)} \quad (4.64)$$

where a new pseudopotential U is defined for the ERTBP, and reads:

$$U = \frac{1}{2} \hat{\omega}^2 (\hat{x}^2 + \hat{y}^2) + \frac{1 - \mu}{\|\hat{r}_1\|} + \frac{\mu}{\|\hat{r}_2\|} \quad (4.65)$$

Equilibrium points can also be identified within the ERTBP, as per Eq. (4.56); however, the new points move with time, and their coordinates should be computed at each time step.

Periodic Motion in the Restricted Three-Body Problem

The dynamics described by Eqs. (4.55) and (4.64) enables the existence of periodic solutions. Nevertheless, periodicity can be found by numerical means only, leveraging an initialization-correction-continuation iterative process.

The initialization is highly tailored to the orbital family to be developed and to the dynamics model. In general, the initial guess can be generated starting from a Lagrangian point, one of the primaries, an analytical approximation, or solutions from a different dynamics model. The output is the initial state of the orbit (or a set of states along the orbit) and its period.

The continuation process consists of introducing a perturbation to a previously defined solution. Various approaches can be leveraged, from a single state element perturbation (“natural parameter continuation”) to methods leveraging local tangent direction to the orbital family (“pseudoarclength continuation”).

The correction process is carried out after the first initialization and after each continuation, to ensure periodicity of the first guess or of the perturbed state, respectively. The correction is based on an iterative, Newton-like numerical approach, where first derivative information of a set of conditions are used to converge to the desired conditions. In particular, periodicity is ensured by achieving the equivalence of the initial state of the orbit and

of the final state. The final state is obtained through a propagation for a certain time, which is an additional variable of the problem, that corresponds to the orbital time after the correction process is converged. The set of conditions reads:

$$\begin{aligned}\mathbf{G}(\boldsymbol{\chi}) &= \phi(\hat{\mathbf{r}}_0, T) - \hat{\mathbf{r}}_0 = 0 \\ \boldsymbol{\chi} &= [\hat{\mathbf{r}}_0^T, T]^T\end{aligned}\quad (4.66)$$

To define the single trajectory within the family, a further condition “ $\boldsymbol{\sigma}(\boldsymbol{\chi})$ ” is required (called “Poincaré phase condition”), which is typically set as a fixed element of the initial state (commonly, the “ x ” or “ z ” component, depending on the specific orbit family) or as a fixed orbital energy. Thus, the full set of conditions reads:

$$\mathbf{F}(\boldsymbol{\chi}) = [\mathbf{G}(\boldsymbol{\chi})^T, \boldsymbol{\sigma}(\boldsymbol{\chi})^T]^T = 0 \quad (4.67)$$

The iterative correction can be performed through the Newton–Raphson formula, as per Eq. (4.68):

$$\boldsymbol{\chi}_{k+1} = \boldsymbol{\chi}_k - [J^T J]^{-1} J^T \mathbf{F}(\boldsymbol{\chi}_k) \quad (4.68)$$

where J is the Jacobian matrix obtained from the derivatives of \mathbf{F} with respect to $\boldsymbol{\chi}$.

The conditions of Eq. (4.67), and the corresponding Jacobian matrix, vary depending on the dynamics model. Such differences are described hereafter.

Circular Restricted Three-Body Problem

The CRTBP is described by an autonomous system. While this ensures the existence of a continuous set of orbits (orbital family), it makes the solution of Eq. (4.67) nonunique, as long as a single Poincaré phase condition is set. The uniqueness is recovered by adding a second condition, which can be again another state element or an energetic constraint.

To build the Jacobian, first the State Transition Matrix (STM) of the problem is required. From the linearization of Eq. (4.55), expressed in the first-order form, the following set is obtained:

$$\begin{aligned}
\mathbf{x} &= \left[\hat{\mathbf{r}}^T, \dot{\hat{\mathbf{r}}}^T \right]^T \\
\dot{\mathbf{x}} &= \mathbf{Ax} \\
\mathbf{A} &= \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 \\ \nabla(\nabla U) & \Omega \end{bmatrix} \\
\Omega &= \begin{bmatrix} 0 & 2 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\end{aligned} \tag{4.69}$$

The STM “ Φ ” is obtained from a time propagation leveraging the matrix \mathbf{A} and setting an identity matrix as initial condition, namely:

$$\begin{aligned}
\dot{\Phi}(t) &= \mathbf{A}\Phi(t) \\
\Phi(0) &= \mathbf{I}_6
\end{aligned} \tag{4.70}$$

Notice that the STM can be propagated in parallel with the state to obtain the 42 elements (6 from the state and 36 from the STM) within a single propagation. Finally, the Jacobian matrix is built as:

$$J = \begin{bmatrix} \Phi(T) & \dot{\phi}(\hat{\mathbf{r}}_0, T) \\ \frac{d\sigma(\chi)}{d\chi} & \mathbf{0} \end{bmatrix} \tag{4.71}$$

A particular situation occurs when orbits around collinear Lagrangian points are developed. In such case, the symmetry with respect to the x - z plane of the synodic frame can be leveraged to reduce the number of variables. According to the *mirror theorem* [14–16], the orbit shall be perpendicular to the aforementioned plane at their intersection points. This applies for both the initial state and for the state at half orbital period. Hence, perpendicularity at half period is imposed by setting:

$$\mathbf{G}(\chi) = \begin{bmatrix} y(T/2) \\ \dot{x}(T/2) \\ \dot{z}(T/2) \end{bmatrix} = 0 \tag{4.72}$$

Since the same conditions apply for the initial state, the problem is solved for the reduced set of variables:

$$\chi = \begin{bmatrix} x_0 \\ z_0 \\ \dot{y}_0 \\ T \end{bmatrix}$$

The corresponding rows and columns of the Jacobian matrix are removed as well.

Elliptic Restricted Three-Body Problem

The ERTBP is characterized by a nonautonomous system, as initial condition of the primaries affects the trajectory of the third body. For this reason, no continuous orbital families can be defined, as only periodic orbits with a commensurable period to the one of the binary systems exist [17,18]. As a consequence, the correction process shall start from an already defined solution in the CRTBP, whose period is resonant with the binary system. Regardless, the nonautonomous nature of the problem requires a single Poincaré condition to uniquely define the periodic solution.

The procedure for defining the correction formula (Eq. 4.68), tailored to the elliptic problem, follows the same steps carried out for the CRTBP case. However, necessary modifications are required to deal with the different dynamics. In particular, the ERTBP dynamics involves a further variable, that is, the eccentric anomaly, as per Eq. (4.64). Hence, Eq. (4.73) shall be extended, and reads:

$$\mathbf{x} = \left[\hat{\mathbf{r}}^T, \dot{\hat{\mathbf{r}}}^T, \dot{E} \right]^T$$

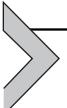
$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_{3 \times 1} \\ \nabla(\nabla U) + \frac{1}{2}\hat{\omega}\Omega & 2\hat{\omega}\Omega & \frac{d\ddot{\mathbf{r}}}{dE} \\ \mathbf{0}_{1 \times 3} & \mathbf{0}_{1 \times 3} & \frac{d\dot{E}}{dE} \end{bmatrix} \quad (4.73)$$

$$\Omega = \begin{bmatrix} 0 & 2 & 0 \\ -2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Here, the additional derivatives of the equations of motion with respect to the eccentric anomaly E (not derived here for the sake of compactness of the expressions) are included in the linearization matrix A .

Regardless, the STM and the Jacobian matrix can be built as per Eqs. (4.70) and (4.71) as in the CRTBP case, keeping in mind that the STM will have one more dimension due to the eccentric anomaly.



Irregular solar system bodies

A large amount of the celestial bodies in the Solar System is represented by asteroids and comets, commonly defined as “small bodies.” Their main characteristics are the very weak gravitational attraction and highly irregular shape. Despite planets and moons are not perfect shapes as well, the gravitational perturbation caused by such irregularities are orders of magnitude below the overall gravity force exerted by the body. On the contrary, local shape variation in small bodies profoundly affect the gravitational field exerted by these objects, and they cannot be neglected. For this reason, several methods and models have been developed to describe irregular gravity fields, and they are becoming very popular in modern GNC applications for missions exploring small celestial objects.

As for the point mass gravitational potential, the derivation of such models starts from the general expression (cfr. Chapter 3 – The Space Environment):

$$U = G \iint_m \frac{1}{r} dm \quad (4.74)$$

being G the gravitational constant, dm the infinitesimal mass portion of the overall attractor, and r the distance between the control point (or space-craft) and such infinitesimal mass.

The difference between each gravitational model derives from the way the integral of Eq. (4.74) is solved.

Spherical Harmonics Expansion Model

The “Spherical Harmonics Expansion (SHE) Model” is a computationally light approach to deal with gravitational irregularities, and it is frequently used to model the perturbations from the small irregularities of planets and moons. Nevertheless, the same approach can be followed to model highly irregular objects.

Following the derivation of the SHE model from [Chapter 3](#) - The Space Environment, the gravitational potential is approximated through *Legendre's Polynomials* [19], leading to the final expression [20]:

$$U_{\text{she}} = \frac{Gm}{r} \left\{ 1 + \sum_{i=2}^n \sum_{j=0}^i \left(\frac{R_0}{r} \right)^i [C_{ij} \cos(j\lambda) + S_{ij} \sin(j\lambda)] P_{ij}(\cos(\theta)) \right\} \quad (4.76)$$

where λ and θ are the attractor's longitude and colatitude, respectively, R_0 a reference radius (typically the average equatorial radius), $P_{ij}(x)$ are “Associated Legendre Polynomials” [21], while C_{ij} and S_{ij} are the normalized Stokes coefficients, computed as:

$$C_{ij} = \frac{1}{m} \sqrt{\frac{(2 - \delta_{0j})(i - j)!}{(2i + 1)(i + j)!}} \iiint_m \left(\frac{r}{R_0} \right)^i P_{ij}(\sin(\theta)) \cos(j\lambda) dm \quad (4.77)$$

$$S_{ij} = \frac{1}{m} \sqrt{\frac{2(i - j)!}{(2i + 1)(i + j)!}} \iiint_m \left(\frac{r}{R_0} \right)^i P_{ij}(\sin(\theta)) \sin(j\lambda) dm$$

Stokes coefficients require the knowledge of the object's shape, which can be reconstructed via preliminary estimation (through ground observations, especially for small solar system bodies), or through in-situ radio-science experiments [22,23] and other techniques [24].

The final accuracy of the gravity field is determined by the number of terms in the expansion from [Eq. \(4.76\)](#). This is particularly critical in the case of small irregular bodies, if compared to the perturbative gravity model of oblate massive bodies. In fact, the low gravity field and the larger irregularities make the higher order terms of the *SHE* more relevant and comparable to the point mass term of the expansion. For this reason, while the sole J_2 term would be sufficient for modeling the perturbation of Earth oblateness, a small asteroid may require contributions above J_{10} , including sectorial and tesseral harmonics (commonly neglected for massive bodies).

Regardless of the higher number of terms, the analytical expression allows a fast evaluation, thus limiting the computational cost.

The real limit of this approach is its applicability only outside the “Brillouin sphere,” the spherical region encompassing the whole attractor, with a radius equal to the maximum surface radius of the body. To deal with gravity modeling inside the sphere, while maintaining a SHE approach, other types of function must be explored, like “Bessel functions” [25].

Ellipsoidal model

In the case the small object can be approximated as a triaxial ellipsoid, an ellipsoidal model (ELL), developed by MacMillan [26], can be exploited. The new formulation has the advantage over the SHE of being valid also within the Brillouin sphere, down to the surface of the object.

By expressing the integral of Eq. (4.74) in spherical coordinates and integrating over the radius, with the ellipsoid formula as upper bound (the reader is invited to consult reference [26] for further details), the new potential reads:

$$U_{\text{ell}} = G\rho\pi abc \int_{\kappa}^{\infty} \frac{1 - \frac{x^2}{a^2+s} - \frac{y^2}{b^2+s} - \frac{z^2}{c^2+s}}{\sqrt{(a^2+s)(b^2+s)(c^2+s)}} ds \quad (4.78)$$

where a, b, c are the three semiaxes, such that $a > b > c$, ρ is the constant density of the body, and the lower bound of the integral, κ , corresponds to the algebraic largest root of equation:

$$\frac{x^2}{a^2 + \kappa} + \frac{y^2}{b^2 + \kappa} + \frac{z^2}{c^2 + \kappa} = 1 \quad (4.79)$$

The evaluation of the integral from Eq. (4.78) leads to the following final expression:

$$U_{\text{ell}} = \frac{2G\rho\pi abc}{\sqrt{a^2 - c^2}} \left\{ \left[1 - \frac{x^2}{a^2 - b^2} + \frac{y^2}{a^2 - b^2} \right] F(\omega_{\kappa}, k) + \left[\frac{x^2}{a^2 - b^2} \right. \right. \\ \left. \left. - \frac{(a^2 - c^2)y^2}{(a^2 - b^2)(b^2 - c^2)} + \frac{z^2}{b^2 - c^2} \right] E(\omega_{\kappa}, k) + \left[\frac{(c^2 + \kappa)y^2}{b^2 - c^2} \right. \right. \\ \left. \left. - \frac{(b^2 + \kappa)z^2}{b^2 - c^2} \right] \frac{\sqrt{a^2 - c^2}}{\sqrt{(a^2 + \kappa)(b^2 + \kappa)(c^2 + \kappa)}} \right\} \quad (4.80)$$

where F and E are the Legendre's elliptic integrals of first and second kind, and ω_{κ} and k satisfy the following conditions:

$$\sin(\omega_{\kappa}) = \sqrt{\frac{a^2 - c^2}{a^2 + \kappa}} \quad (4.81)$$

$$k^2 = \frac{a^2 - b^2}{a^2 - c^2}$$

The accuracy of this model is strongly dependent on how close the actual body surface is to a triaxial ellipsoid. It is worth mentioning that the evaluation of Eq. (4.80) is typically computationally heavier than the SHE model due to the computation of the elliptic integrals. Hence, such model is suggested only if the spacecraft is very close or inside the Brillouin sphere.

Mass concentration model

The principle behind the Mass Concentration (MasCon) differs from the previous approaches, as the integral of Eq. (4.74) is not solved continuously through some approximation, but rather it is discretized and solved as a summation of many point masses composing the attractor. The result is a cluster of points that take, overall, the actual shape of the body.

The expression of the MasCon potential reads:

$$U_{\text{MasCon}} = G \sum_{i=1}^{N_m} \frac{m_i}{r_i} \quad (4.82)$$

where m_i and r_i are respectively the mass of the i th point mass of the cluster, and its distance from the field point where the potential is being evaluated. N_m is the total number of masses that approximate the body.

The expression of Eq. (4.82) is simpler than the other methods that solve the continuous integral. However, the difficulty of the MasCon approach lies in the strategy for distributing the masses within the attractor's volume. Indeed, the challenge is to find a balance between the total number of masses and their distribution.

In general, as the number of masses increases, the accuracy of the potential estimation improves, but the numerical effort to compute the potential value becomes higher. If their distribution is well arranged, it is possible to maintain the same accuracy level with less masses, thus improving the computational performances. Several strategies for the distribution of masses have been developed, and include “uniform distributions,” “multiple cores,” “multiple layers,” “core-shell,” and others [27].

The great advantage of such model is the capability of dealing with possible internal cavities or changes of density within the body. Nevertheless, the computational burden is relatively high, plus field points close to surface suffer from oscillations of the potential, due to the singularities introduced by the single point masses [28].

Polyhedral model

The “Polyhedral Model” (POL) is a method developed by Werner and Scheeres [29], which solves the integral of the potential expression by modeling the attractor as a set of tetrahedrons, making the overall body a polyhedral object. The tetrahedrons have always one vertex coincident with the center of mass of the body and the opposite face representing the surface element of the body.

After several rearrangements of the integral expression of Eq. (4.74), leveraging normal and tangent vectors to the external surface of the tetrahedrons (the reader is invited to consult Ref. [29] for the formal derivation), the final potential expression reads:

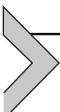
$$U_{\text{poly}}(x, y, z) = -\frac{1}{2} G \rho \left(\sum_{f \in \text{faces}} r_f \cdot F_f \cdot r_f \omega_f - \sum_{e \in \text{edges}} r_e \cdot E_e \cdot r_e L_e \right) \quad (4.83)$$

F_f is the dyad associated to face f , and E_e is the dyad associated to edge e of the polyhedron model, and read:

$$\begin{aligned} F_f &= \hat{n}_f \hat{n}_f \\ E_e &= \hat{n}_{f_1} \hat{n}_e^f + \hat{n}_{f_2} \hat{n}_e^f \end{aligned} \quad (4.84)$$

where f_1 and f_2 denote the two faces sharing the edge e . The term L_e represents the potential of a wire associated to the edge e and depends on the distance between the field point and the edge’s ends and on the edge’s length. ω_f is the solid angle associated to the face f , and it is dependent on the distance vectors between the field point and the three face’s vertexes.

The accuracy of the POL is uniquely related to the number of faces that constitute the full polyhedron. The great advantage of such model is the correctness of the gravitational field down to the surface of the body, without any restriction in the shape of the body itself (as it is for the ELL model). The downside is the high computational burden given by the summations in Eq. (4.83), especially when the number of tetrahedrons composing the body becomes very large [28,30].



Relative orbital dynamics

The term relative dynamics generally refers to the description of the motion of one body with respect to a moving reference frame. In this book, relative orbital dynamics strictly refers to the equations of motion of a spacecraft with respect to a free-falling nonattractive point, material or fictitious, in orbit.

Such dynamical expression is quite common in missions involving rendezvous and docking, formation-flying, on-orbit servicing, and proximity operations, which are common applications in modern spacecraft GNC. To obtain impulsive maneuvers and acceleration analysis from observations made in the non-inertial frame, we need to transform them into an inertial frame. Otherwise, it is impossible to distinguish between thrusting forces and inertia forces. The aim of this chapter is to give the critical insights on the derivation of spacecraft relative orbital dynamics, but it assumes that the reader is familiar with Newtonian mechanics and vector derivatives. A brief discussion on relative spacecraft attitude dynamics is included in [Chapter 5](#) – Attitude Dynamics.

Let us assume that we want to describe the motion of a spacecraft and to attach a reference frame $\Delta_{i,j,k}$ (typically the adopted comoving reference frame is the Local-Vertical-Local-Horizontal [LVLH]) to an orbiting target or a fictitious point that follows an orbital trajectory with angular velocity $\Omega(t)$. In this derivation, the inertial reference frame is referred as $\mathbb{I}_{J,K}$. Please note that it represents the same inertial ECI reference frame presented in [Chapter 2](#) – Reference Systems and Planetary Models. Hereby, we recall useful relationship to describe the angular velocity and its derivative of an orbital point referred with subscript 0:

$$\begin{aligned}\Omega &= \frac{\mathbf{h}}{r_0^2} = \frac{\mathbf{r}_0 \times \mathbf{v}_0}{r_0^2} \\ \dot{\Omega} &= -2 \frac{h}{r_0^3} \dot{\mathbf{r}}_0 = -2 \frac{\mathbf{v}_0 \cdot \mathbf{r}_0}{r_0^2} \Omega\end{aligned}$$

With reference to [Fig. 4.4](#), we can define the following vectorial quantities. First, the inertial position of the spacecraft in the inertial frame can be described as:

$$\mathbf{r}_{\mathbb{I}} = \mathbf{r}_{0,\mathbb{I}} + \delta \mathbf{r}_{\mathbb{I}} \quad (4.85)$$

The unit vectors of the $\Delta_{i,j,k}$ reference frame can be defined as:

$$\hat{\mathbf{i}} = \frac{\mathbf{r}_0}{r_0}, \quad \hat{\mathbf{k}} = \frac{\mathbf{h}}{h}, \quad \hat{\mathbf{j}} = \hat{\mathbf{k}} \times \hat{\mathbf{i}}$$

where \mathbf{h} is the specific angular momentum. This allows us to transform the relative position and velocity into the comoving frame as:

$$\delta \mathbf{r}_{\Delta} = \delta x \hat{\mathbf{i}} + \delta y \hat{\mathbf{j}} + \delta z \hat{\mathbf{k}}$$

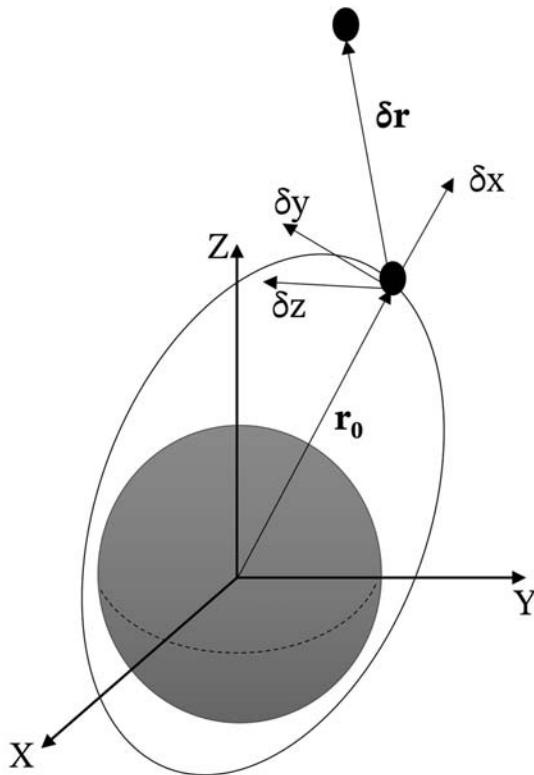


Figure 4.4 Moving frame for relative motion description.

$$\delta \mathbf{v}_\Delta = \dot{\delta x} \hat{\mathbf{i}} + \dot{\delta y} \hat{\mathbf{j}} + \dot{\delta z} \hat{\mathbf{k}}$$

Taking advantage of the topic discussed in the previous section describing the two-body problem, we have the characteristic R2BP equation:

$$\ddot{\mathbf{r}}_{\parallel} = -\frac{Gm_1}{\mathbf{r}^3} \mathbf{r}_{\parallel} = -\frac{\mu}{r^3} \mathbf{r}_{\parallel} \quad (4.86)$$

By recalling Eqs. (4.84 and 4.85), one can simply substitute to get the explicit dependence of the relative position:

$$\ddot{\delta \mathbf{r}}_{\parallel} = -\ddot{\mathbf{r}}_{0,\parallel} - \frac{\mu}{r^3} (\mathbf{r}_{0,\parallel} + \delta \mathbf{r}_{\parallel}) \quad (4.87)$$

So far, we have developed equations with vector quantities referred to the inertial frame \parallel . Nevertheless, as already mentioned, we are interested in formulating the equations of motion with respect to a moving frame.

To this purpose, the classical relative kinematics solves our issues. Indeed, the relative acceleration measured in the comoving frame can be expressed as:

$$\delta \mathbf{a}_\Delta = \ddot{\delta x} \hat{\mathbf{i}} + \ddot{\delta y} \hat{\mathbf{j}} + \ddot{\delta z} \hat{\mathbf{k}}$$

which is related to the absolute relative acceleration by the following relation:

$$\delta \ddot{\mathbf{r}}_\Delta = \delta \mathbf{a}_\Delta + \dot{\Omega} \times \delta \mathbf{r}_\Delta + \Omega \times (\Omega \times \delta \mathbf{r}_\Delta) + 2\Omega \times \delta \mathbf{v}_\Delta \quad (4.88)$$

where the relative acceleration measured in the moving frame is added to the contributions known as Euler acceleration $\dot{\Omega} \times \delta \mathbf{r}_\Delta$, centrifugal acceleration $\Omega \times (\Omega \times \delta \mathbf{r}_\Delta)$, and Coriolis acceleration $2\Omega \times \delta \mathbf{v}_\Delta$. Now, if we express Eq. (4.87) into the moving frame, knowing that $\mathbf{r}_{0,\Delta} = r_0 \hat{\mathbf{i}}$ by definition, and substitute Eq. (4.88), we obtain the nonlinear expression of the relative dynamics in the moving frame:

$$\left\{ \begin{array}{l} \delta \ddot{x} - 2\Omega \delta \dot{y} - \dot{\Omega} \delta y - \Omega^2 \delta x = -\mu \frac{(r_0 + \delta x)}{[(r_0 + \delta x)^2 + \delta y^2 + \delta z^2]^{\frac{3}{2}}} + \frac{\mu}{r_0^2} \\ \delta \ddot{y} + 2\Omega \delta \dot{x} + \dot{\Omega} \delta x - \Omega^2 \delta y = -\mu \frac{\delta y}{[(r_0 + \delta x)^2 + \delta y^2 + \delta z^2]^{\frac{3}{2}}} \\ \delta \ddot{z} = -\mu \frac{\delta z}{[(r_0 + \delta x)^2 + \delta y^2 + \delta z^2]^{\frac{3}{2}}} \end{array} \right. \quad (4.89)$$

One can immediately perceive some key insights of the unperturbed relative motion: in general, the moving reference frame position is a function of the time $r_0 = r_0(t)$, as well as the angular velocity $\Omega = \Omega(t)$. It can be demonstrated that there exists only one equilibrium point that coincides with the origin of the moving reference frame $\exists! \delta \mathbf{r}_{eq} = 0$.

Hence, we actually need additional differential equations in order to solve the motion. In the following paragraphs, some assumptions will be made in order to converge to analytical solutions, which are of paramount importance in GNC design.

Linearization of the equations of motion

The equations of motion described in Eq. (4.89) are nonlinear, and there exists no analytical solution to predict the motion. Obviously, the equations can be solved numerically, provided that the motion of the moving frame is known, or at least the differential equations describing it. What we strive to obtain

during GNC design is an analytical solution in order to derive and synthesize the algorithms. Often, the analytical solutions provide good accuracies under specific assumptions. It is safe to state that, although we may think of implementing more accurate models, analytical solutions are critical to investigate the expected motion: we need them to get a better insight on the motion evolution without using an immense pool of numerical simulations. A schematic of the crucial steps toward simplifying the model is reported in Fig. 4.5.

The first step toward this goal is to linearize the system of ordinary differential equations presented in Eq. (4.88). If we set the assumption of close relative motion, we can state that:

$$\frac{\delta r}{r_0} \ll 1$$

This means that we want to describe the relative motion of two objects in orbit, whose relative distance is much less than their distance with respect to the attracting body. Such assumption sounds very reasonable if we think of the typical applications in which relative dynamics is used, for instance, rendezvous and proximity operations. Thus, making use of Taylor expansion and Eqs. (4.84a and b), we can write:

$$\delta \ddot{\mathbf{r}}_{\parallel} \approx -\frac{\mu}{r_0^3} \left[\delta \mathbf{r}_{\parallel} - \frac{3}{r_0^2} (\mathbf{r}_{0,\parallel} \cdot \delta \mathbf{r}_{\parallel}) \mathbf{r}_0 \right] \quad (4.90)$$

where all the terms in $\frac{\delta r}{R}$ higher than order one have been neglected. Similar to the nonlinear derivation, taking Eq. (4.88) and substituting in Eq. (4.90), we obtain the set of linear second-order differential equations describing the relative motion expressed in the comoving frame (LVLH):

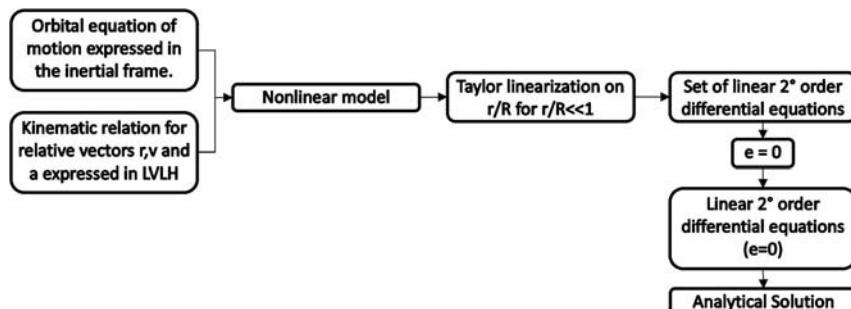


Figure 4.5 Schematics of relative dynamics derivation and main assumptions.

$$\left\{ \begin{array}{l} \delta\ddot{x} - \left(\frac{2\mu}{r_0^3} + \frac{h^2}{r_0^4} \right) \delta x + \frac{2(\mathbf{v}_0 \cdot \mathbf{r}_0)h}{r_0^4} \delta y - \frac{2h}{r_0^2} \delta \dot{y} = 0 \\ \delta\ddot{y} + \left(\frac{\mu}{r_0^3} - \frac{h^2}{r_0^4} \right) \delta y - \frac{2(\mathbf{v}_0 \cdot \mathbf{r}_0)h}{r_0^4} \delta x + \frac{2h}{r_0^2} \delta \dot{x} = 0 \\ \delta\ddot{z} + \frac{\mu}{r_0^3} \delta z = 0 \end{array} \right. \quad (4.91)$$

The linearized equations of motion show some key insights on the relative motion in close proximity:

- The in-plane components, namely δx and δy , are coupled, whereas the cross-track component is independent. Moreover, the cross-track component presents the peculiar form of the harmonic oscillator.
- As for the nonlinear version, the vectors \mathbf{r}_0 and \mathbf{v}_0 are generally functions of time. The two vectors represent the position and velocity of the comoving reference frame center, which typically lies on the path of a reference orbit. For instance, for eccentric reference orbits, \mathbf{r}_0 and \mathbf{v}_0 vary with time, although with constant specific angular momentum.

Similarly, for the linearized equations, there is not an easy analytical solution given the time dependence of the reference frame position and velocity in the linear differential equations. The derivation so far is the most general one could work with in presence of the unperturbed relative motion.

Many researchers have studied alternative forms and parametrizations to be able to solve the set of linear differential equations. Hereby, we briefly discuss the most used models in GNC design.

True anomaly parametrization in linearized relative dynamics

One strategy to work out analytical solutions is to reparametrize the time dependence as a function of the true anomaly. In other words, such approach allows to solve for the relative motion of the spacecraft as a function of the true anomaly of the reference orbit. The true anomaly is indeed used as free-variable. The time derivative (\cdot) of a given quantity is linked to the derivative $(\cdot)'$ with respect to the true anomaly in this way:

$$(\cdot) = (\cdot)' \dot{\theta}$$

$$(\cdot) = (\cdot)'' \dot{\theta}^2 + \dot{\theta} \dot{\theta}' (\cdot)'$$

Using fundamentals notions of two-body orbital mechanics, described in the previous chapter, we can express the reference orbit quantities as:

$$r_0(\theta) = \frac{a(1-e^2)}{1+e\cos\theta}; \quad \Omega(\theta) = \frac{n(1+e\cos(\theta))^2}{(1-e^2)^{\frac{3}{2}}}; \quad n = \left(\frac{\mu}{a^3}\right)^{\frac{1}{2}}$$

With these transformations, the set of linear time-varying second-order differential equations arising from Eq. (4.90) can be rewritten for a generic eccentric reference orbit:

$$\begin{cases} \delta x'' = \frac{2e\sin(\theta)}{1+e\cos(\theta)}\delta x' + \frac{3+e\cos(\theta)}{1+e\cos(\theta)}\delta x + 2\delta y' - \frac{2e\sin(\theta)}{1+e\cos(\theta)}\delta y \\ \delta y'' = -2\delta x' + \frac{2e\sin(\theta)}{1+e\cos(\theta)}\delta x + \frac{2e\sin(\theta)}{1+e\cos(\theta)}\delta y' - \frac{e\cos(\theta)}{1+e\cos(\theta)}\delta y \\ \delta z'' = \frac{2e\sin(\theta)}{1+e\cos(\theta)}\delta z' - \frac{1}{1+e\cos(\theta)}\delta z \end{cases} \quad (4.92)$$

The equation presents the same peculiarities on the in-plane versus out-of-plane components decoupling. The difficulty results from the linearization process, which maps the curvilinear space to a rectangular one by a small curvature approximation. Fig. 4.6 shows the effects of the linearization and the small curvature assumption. In this case, a relative separation in the in-track direction in the linearized equations corresponds to an incremental phase difference in true anomaly.

Eq. (4.92) can be solved analytically in its homogenous form to obtain the solution for the motion. The solutions are available in the literature in various forms using different reference frames and variables. The most notable solutions are presented by Carter [31], Inshan and Tillerson [32], which extend Carter's work. However, the analytical solutions, and

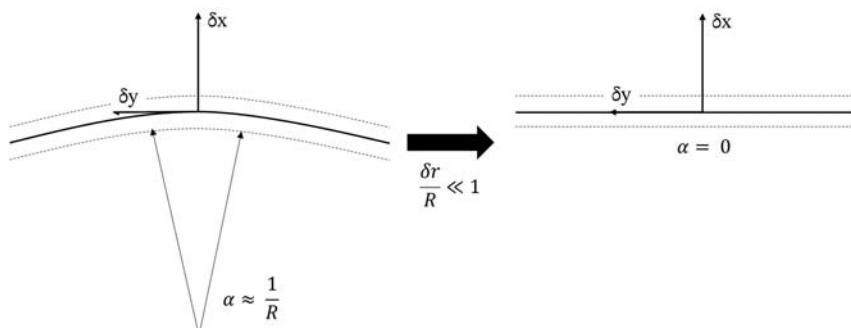


Figure 4.6 Mapping from curvilinear to linear space.

consequent STM, imply nontrivial integrations: this makes the formulation complex for engineering use.

A remarkable solution and STM has been proposed by Yamanaka and Ankersen [33]. If we set the following coordinate transformation, calling $\rho = (1 + e \cos(\theta))$:

$$\begin{pmatrix} \delta\tilde{x} \\ \delta\tilde{\gamma} \\ \delta\tilde{z} \end{pmatrix} = \rho \begin{pmatrix} \delta x \\ \delta \gamma \\ \delta z \end{pmatrix}$$

Whose derivative with respect to the true anomaly is in the form of:

$$\delta\tilde{x}' = \rho\delta x' - e \sin(\theta)\delta x$$

$$\delta\tilde{x}'' = \rho\delta x'' - 2e \sin(\theta)\delta x' - e \cos(\theta)\delta x$$

Thus, Eq. (4.92) can be rewritten simply as:

$$\left\{ \begin{array}{l} \delta\tilde{x}'' = \frac{3\delta\tilde{x}}{\rho} + 2\delta\tilde{\gamma} \\ \delta\tilde{\gamma}'' = -2\delta\tilde{x}' \\ \delta\tilde{z}'' = -\delta\tilde{z} \end{array} \right. \quad (4.93)$$

Eq. (4.93) can be solved by using the proposed integral formulation [33] to derive the following solution algorithm for the STM, which is very much used in GNC algorithm design. In the original formulation of Yamanaka, the comoving frame $\boldsymbol{\Gamma}_{m,n,p}$ is taken differently from $\Delta_{i,j,k}$, in particular, the following rotation holds from $\boldsymbol{\Gamma}_{m,n,p}$ to $\Delta_{i,j,k}$:

$$\varDelta \mathbf{R}_\Gamma = \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}.$$

In practical words, the radial direction toward the attracting body is called z axis, whereas the γ axis is opposite to the angular momentum vector.

1. Given a set of initial conditions, we find the transformed variables:

$$\delta\tilde{\mathbf{r}}_0 = \rho \delta\mathbf{r}_0; \quad \delta\tilde{\mathbf{v}}_0 = -e \sin(\theta) \delta\mathbf{r}_0 + \left(\frac{p^2}{h\rho} \right) \delta\mathbf{v}_0$$

2. The pseudoinitital values for the in-plane components are necessary in order to use the proposed STM. They need to be calculated as follows, bearing in mind that they are expressed in the $\Gamma_{m,n,p}$ reference frame:

$$\begin{pmatrix} \delta\tilde{x}_0 \\ \delta\tilde{z}_0 \\ \delta\tilde{v}_{x0} \\ \delta\tilde{v}_{z0} \end{pmatrix} = \frac{1}{1-e^2} \begin{bmatrix} 1 - e^2 & 3es(1/\rho + 1/\rho^2) & -es(1 + 1/\rho) & -e\dot{c} + 2 \\ 0 & -3s(1/\rho + e^2/\rho^2) & s(1 + 1/\rho) & c - 2e \\ 0 & -3(c/\rho + e) & c(1 + 1/\rho) + e & -s \\ 0 & 3\rho + e^2 - 1 & -\rho^2 & -es \end{bmatrix} \begin{pmatrix} \delta\tilde{x}_0 \\ \delta\tilde{z}_0 \\ \delta\tilde{v}_{x0} \\ \delta\tilde{v}_{z0} \end{pmatrix}$$

where $c = \rho \cos(\theta)$ and $s = \rho \sin(\theta)$. The out-of-plane (or cross-track) initial conditions can be used as they are in the STM.

3. The STM for the in-plane and out-of-plane reads, respectively:

$$\begin{pmatrix} \delta\tilde{x}_t \\ \delta\tilde{z}_t \\ \delta\tilde{v}_{xt} \\ \delta\tilde{v}_{zt} \end{pmatrix}_{\Gamma} = \begin{bmatrix} 1 & -c(1 + 1/\rho) & -s(1 + 1/\rho) & 3\rho^2 J \\ 0 & s & c & 2 - 3esJ \\ 0 & 2s & 2c - e & 3(1 - 2esJ) \\ 0 & s' & c' & -3e(s'J + s/\rho^2) \end{bmatrix}_{\theta} \begin{pmatrix} \delta\tilde{x}_0 \\ \delta\tilde{z}_0 \\ \delta\tilde{v}_{x0} \\ \delta\tilde{v}_{z0} \end{pmatrix}$$

$$\begin{pmatrix} \delta\tilde{\gamma}_t \\ \delta\tilde{v}_{\gamma t} \end{pmatrix}_{\Gamma} = \frac{1}{\rho|_{\theta-\theta_0}} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}_{\theta-\theta_0} \begin{pmatrix} \delta\tilde{\gamma}_0 \\ \delta\tilde{v}_{\gamma 0} \end{pmatrix}$$

where θ can be calculated at any time using Kepler's equation and the auxiliary terms are:

$$\rho = 1 + e \cos(\theta), \quad s = \rho \sin(\theta), \quad c = \rho \cos(\theta)$$

$$\begin{aligned} s' &= \cos(\theta) + e \cos(2\theta), \quad c' = -(\sin(\theta) + e \sin(2\theta)), \quad J = k^2(t - t_0), \quad k^2 \\ &= h/p^2 \end{aligned}$$

Linearized equations of motion for nearly circular orbits

The particular case in which the reference orbit can be assumed circular (or nearly circular), the equations of motion can be further simplified and deeply treated analytically. Such assumption can be formalized as:

$$e = 0 \rightarrow \mathbf{v}_0 \cdot \mathbf{r}_0 = 0 \rightarrow h = \sqrt{\mu r_0} \rightarrow n = \sqrt{\frac{\mu}{r_0^3}}$$

where n is called mean motion. In a circular orbit, the position and velocity vector are always perpendicular. The moving reference frame rotates around the attracting body with a constant angular velocity, n indeed. The resultant linearized equations of motion for nearly circular orbits are best known as Clohessy–Wiltshire (C–W) model and read (in the Δ frame):

$$\begin{cases} \delta\ddot{x} - 3n^2\delta x - 2n\delta\dot{y} = 0 \\ \delta\ddot{y} + 2n\delta\dot{x} = 0 \\ \delta\ddot{z} + n^2\delta z = 0 \end{cases} \quad (4.94)$$

Since there are no time-dependent coefficients, the equations can be integrated to derive the analytical solutions for the set of initial conditions $\delta\mathbf{r}_{0\Delta} = (\delta x_0, \delta y_0, \delta z_0, \delta v_{x0}, \delta v_{y0}, \delta v_{z0})$. The solutions are here reported in the state-space form as STM, which is very useful for the purpose of GNC design:

$$\begin{pmatrix} \delta x_t \\ \delta y_t \\ \delta z_t \\ \delta v_{x0} \\ \delta v_{y0} \\ \delta v_{z0} \end{pmatrix}_{\Delta} = \begin{bmatrix} 4 - 3 \cos(nt) & 0 & 0 & \frac{1}{n} \sin(nt) & \frac{2}{n} (1 - \cos(nt)) & 0 \\ 6(\sin(nt) - nt) & 1 & 0 & \frac{2}{n}(\cos(nt) - 1) & \frac{1}{n}(4 \sin(nt) - 3nt) & 0 \\ 0 & 0 & \cos(nt) & 0 & 0 & \frac{1}{n} \sin(nt) \\ 3n \sin(nt) & 0 & 0 & \cos(nt) & 2 \sin(nt) & 0 \\ 6n(\cos(nt) - 1) & 0 & 0 & -2 \sin(nt) & 4 \cos(nt) - 3 & 0 \\ 0 & 0 & -n \sin(nt) & 0 & 0 & \cos(nt) \end{bmatrix}_t \begin{pmatrix} \delta x_0 \\ \delta y_0 \\ \delta z_0 \\ \delta v_{x0} \\ \delta v_{y0} \\ \delta v_{z0} \end{pmatrix}_{\Delta} \quad (4.95)$$

The solutions are truly insightful to derive some peculiar features of the relative motion:

- As already seen, the in-plane components, namely δx and δy , are coupled, whereas the cross-track component is independent. Moreover, the cross-track component presents the peculiar form of the harmonic oscillator.
- All the components are a composition of sinusoidal functions with $1/n$ as fundamental frequency.
- The radial and cross-track components are purely harmonic.
- The only component that possesses a secular term is the along-track component δz .

The C-W equations are very relevant for control and guidance synthesis. Furthermore, they provide a key tool to investigate the relative motion and its geometry with simple considerations on the initial conditions. The drawbacks of such model are the restricting assumptions that involve both a boundary on eccentricity ($e \rightarrow 0$) and on the relative position. Roughly speaking, a general guideline for the applicability of this model in one period prediction, we could set as upper boundaries: $\frac{\delta x}{r_0} < 8e^{-3}$, $\frac{\delta y}{r_0} < 6e^{-2}$, $\frac{\delta z}{r_0} < 6e^{-2}$.

Analysis and characteristic of the unperturbed motion

The solutions of the linearized nearly circular motion allow a deep understanding of the relative motion. In particular, we could manipulate the equations to derive the fundamental characteristic of the unperturbed relative motion. We already know that the cross-track component is completely decoupled from the in-plane components. The out-of-plane motion follows harmonic oscillations around the origin. To analyze the in-plane motion, let us recast the solutions in Eq. (4.95) as:

$$\delta x = C_1 + C_4 \sin(nt) - C_3 \cos(nt) \quad (4.96a)$$

$$\delta y = C_2 - \frac{3}{2}C_1 nt + 2C_3 \sin(nt) + C_4 \cos(nt) \quad (4.96b)$$

$$\text{where } C_1 = \left(4\delta x_0 + \frac{2\delta v_{y0}}{n} \right), \quad C_2$$

$$= \left(\delta y_0 - 2\frac{\delta v_{x0}}{n} \right), \quad C_3 = \left(3\delta x_0 + 2\frac{\delta v_{y0}}{n} \right), \quad C_4 = \frac{\delta v_{x0}}{n}.$$

If we take the linear combination of the squared Eqs. (4.96a and b), we obtain:

$$\begin{aligned} & (\delta x - C_1)^2 + \frac{\left(\delta y - C_2 + \frac{3}{2}C_1(nt)\right)^2}{4} \\ &= C_3^2 + C_4^2 \text{ if } (C_3^2 + C_4^2) \neq 0 \text{ then } \frac{(\delta x - C_1)^2}{(C_3^2 + C_4^2)} \\ &+ \frac{\left(\delta y - C_2 + \frac{3}{2}C_1(nt)\right)^2}{4(C_3^2 + C_4^2)} = 1 \end{aligned}$$

The latter formulation resembles the canonical formulation of an ellipse in the orbital plane. In particular, when $(C_3^2 + C_4^2) \neq 0$, the equation describes an ellipse in δx , δy plane, whose δy center coordinates vary linearly in time. The center coordinates are given as:

$$\begin{aligned} \delta x_C &= 4\delta x_0 + 2\frac{\delta v_{y0}}{n} \\ \delta y_C &= \delta y_0 - 2\frac{\delta v_{x0}}{n} - \frac{3}{2}\left(4\delta x_0 + 2\frac{\delta v_{y0}}{n}\right)nt \rightarrow \delta \dot{y}_C \\ &= -3(2n\delta x_0 + \delta v_{y0}) \end{aligned}$$

In general, the semimajor axis is along the y -axis with a value of $a = 2\sqrt{(C_3^2 + C_4^2)}$ and semiminor axis $b = \frac{a}{2}$ meaning that the amplitude of the along-track relative motion is twice the radial one. The interesting result is that, regardless of the initial condition, the relative elliptical orbits (osculating ellipses whose center varies) have always an eccentricity of $e = \frac{1}{2}\sqrt{3}$. The generic drifting motion is shown in Fig. 4.7.

Particular cases can be identified by inspecting Eq. (4.95) and the following derivations. In particular, the motion can be investigated by working out relationships in terms of the initial conditions. For the unperturbed and unforced motion, the initial conditions fully determine the motion. Initial conditions on the relative velocities different from zero can be thought as the results of impulsive shots.

Concentric coplanar absolute orbit

If we investigate the term $C_3^2 + C_4^2$ and set it to 0, we no longer obtain a moving ellipse. In detail, by removing the assumption adopted beforehand,

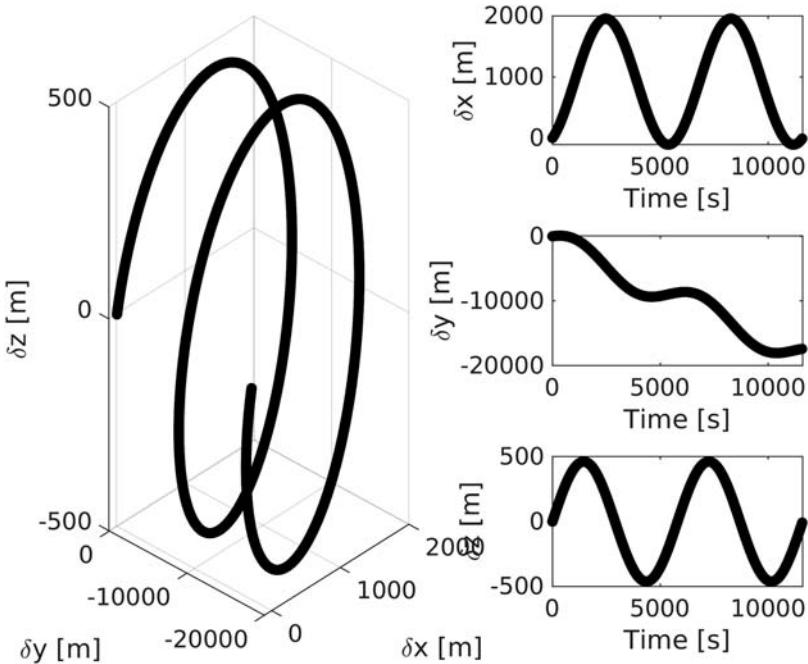


Figure 4.7 Generic relative motion. Drifting trajectories: the in-plane projection is a set of osculating ellipses.

we can set $C_3 = C_4 = 0$, which is the only condition that vanishes the term $C_3^2 + C_4^2$. In addition, taking advantage of the decoupling between the in-plane and out-of-plane motion, we focus on the coplanar scenario. Summarizing the initial conditions constraints:

$$\delta v_{x0} = 0; \delta v_{y0} = -\frac{3}{2}n\delta x_0; \delta z_0 = 0; \delta v_{z0} = 0$$

We obtain a spacecraft that orbits the attracting body in a circular orbit, close to the reference one. The circular orbit has a slight difference in radius given by the entity of δx_0 . If for instance, if the circular orbit has a slightly larger radius ($\delta x_0 > 0$), then the spacecraft simply lags the reference frame. The plot in Fig. 4.8 shows a straight line due to the linearization of the analyzed model.

Circular relative orbit

A relative circular motion around the moving reference frame origin can be achieved by canceling the secular term. If we set $C_1 = C_2 = 0$, we obtain a

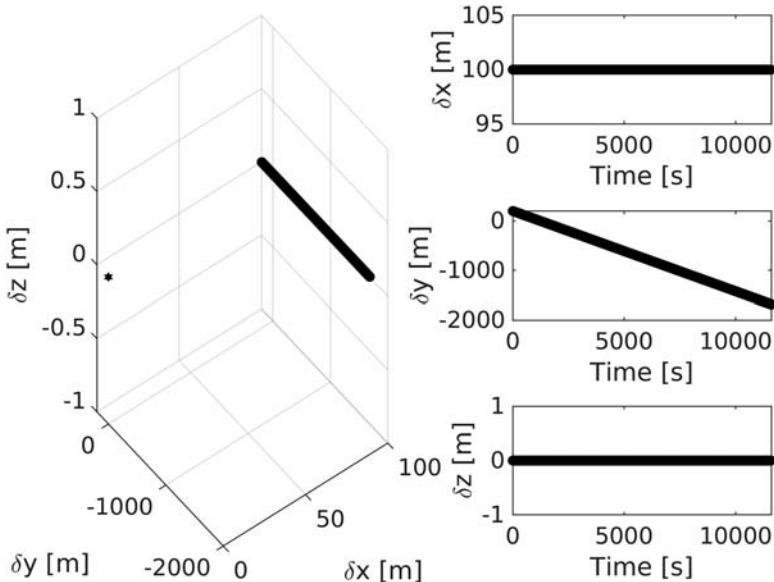


Figure 4.8 Relative motion from concentric coplanar absolute orbits.

centered relative orbit. For the above conditions the center offset, and the drifting term, vanish. The in-plane motion is an ellipse, centered at the origin of the moving reference frame, with the semimajor axis lying on the along-track direction and the semiminor axis along the radial direction, as shown in Fig. 4.9.

Stationary coplanar elliptical relative orbit

The center of the relative ellipse does not necessarily need to be at the origin of the moving reference frame. Hence, if we only keep the assumption of $C_1 = 0$, then the motion takes place in the orbital plane of the reference orbit. Consequently, the motion of the spacecraft is an ellipse, whose center does not move with respect to the origin of the moving reference frame, as shown in Fig. 4.10.

Impulsive shots

An initial condition in the relative velocities can be regarded as an impulsive shot, i.e., an instantaneous change of velocity from the equilibrium point. The resultant motion can be summarized as in Table 4.3.

The impulsive relative maneuvers will be described in detail in Chapter 8—Guidance.

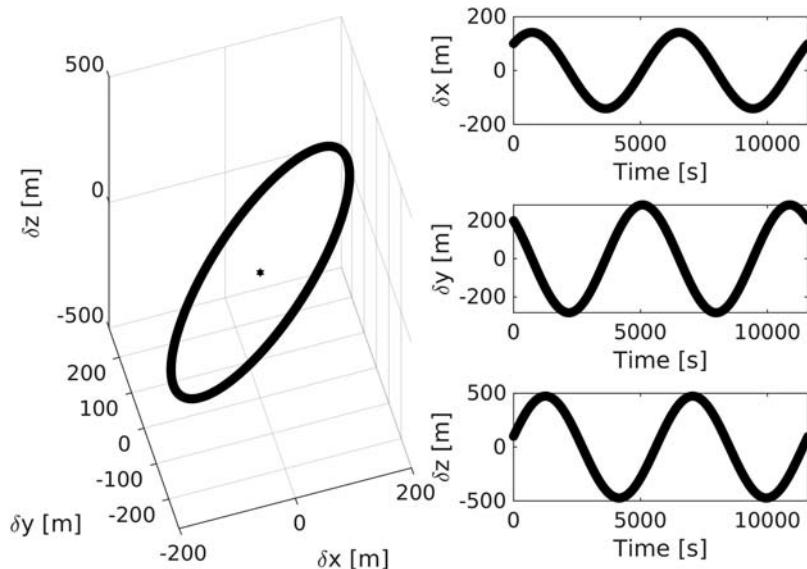


Figure 4.9 Three-dimensional circular relative orbit.

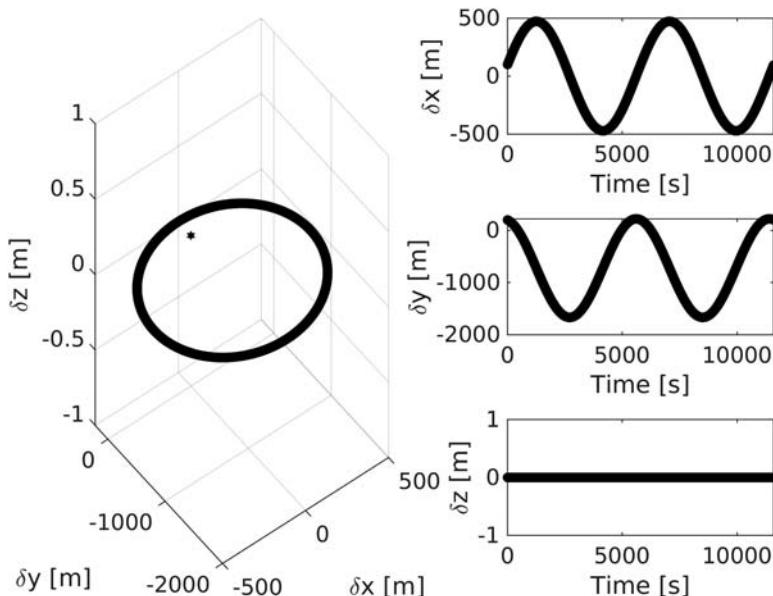


Figure 4.10 Stationary coplanar elliptical relative orbit.

Table 4.3 Impulsive motion in linearized relative dynamics for nearly circular orbits.

Impulse direction	Motion description
Along-track	$\delta\gamma$ exhibits a periodic variation superimposed with a linear drift
Radial	Purely periodic in δx , δy . Bounded and closed relative orbit without any drift.
Normal	Cross-track oscillatory motion.

J_2 -perturbed relative dynamics

Several authors have proposed relative dynamical models that include perturbations. Certainly, one of the most important perturbations is the J_2 -term due to Earth oblateness. Here, we report the formulation of the J_2 -perturbed relative dynamics without reporting the detailed derivation, which can be found in Ref. [34]. The nonlinear J_2 -perturbed dynamics reads:

$$\begin{aligned}\delta\ddot{x} = & 2\omega_z\delta\dot{y} - \left(n_j^2 - \omega_z^2\right)\delta x + \alpha_z\delta\gamma - \omega_x\omega_z\delta z - (\zeta_j - \zeta)s_is\theta \\ & - r\left(n_j^2 - n^2\right) + a_x\end{aligned}$$

$$\begin{aligned}\delta\ddot{y} = & -2\omega_z\delta\dot{x} + 2\omega_x\delta z - \alpha_z\delta x - \left(n_j^2 - \omega_z^2 - \omega_x^2\right)\delta y + \alpha_x\delta z \\ & - (\zeta_j - \zeta)s_ic\theta + a_y\end{aligned}$$

$$\delta\ddot{z} = -2\omega_x\delta\dot{y} - \omega_x\omega_z\delta x - \alpha_x\delta y - \left(n_j^2 - \omega_x^2\right)\delta z - (\zeta_j - \zeta)c_i + \alpha_z + a_z$$

where the contributing terms are:

$$n^2 = \frac{\mu}{r_0^3} + \frac{k_{J2}}{r_0^5} - \frac{5k_{J2}s_i^2s_\theta^2}{r_0^5}, \quad \eta_{JZ} = (r_0 + \delta x)s_is\theta + \delta ys_is\theta + \delta zc_i$$

$$n_j^2 = \frac{\mu}{r^3} + \frac{k_{J2}}{r^5} - \frac{5k_{J2}r_{JZ}^2}{r^7}, \quad k_{J2} = \frac{3J_2\mu R_e^2}{2}$$

$$\omega_x = -\frac{k_{J2}s_is\theta}{hr_0^3}, \quad \omega_z = \frac{h}{r_0^2}$$

$$\alpha_x = \dot{\omega}_x = \frac{k_{J2}s_{2i}c_0}{r_0^5} + \frac{3\dot{r}_0k_{J2}s_{2i}s\theta}{r_0^4 h} - \frac{8k_{J2}^2 s_i^3 c_i s_\theta^2 c_0}{r^6 h^2}$$

$$\alpha_z = \dot{\omega}_z = -\frac{2h\dot{r}_0}{r_0^3} - \frac{k_{J2}s_i^2 s_{2\theta}}{r_0^5}, \quad \zeta = \frac{2k_{J2}s_i s_\theta}{r^4}, \quad \zeta_j = \frac{2k_{J2}r_{JZ}}{r^5}$$

in which h is the orbital angular momentum, i orbital inclination, J_2 is the zonal harmonic coefficient $1.0826 \cdot 10^{-3}$ for Earth, R_e is the Earth radius, θ is the orbital true anomaly, and $a = [a_x, a_y, a_z]^T$ is the forced acceleration vector. Last, s_x and c_x stand for $\sin(x)$ and $\cos(x)$, where x is a generic angle. The spacecraft relative motion is actually described by 11 first-order differential equations, namely $(\delta x, \delta y, \delta z, \delta \dot{x}, \delta \dot{y}, \delta \dot{z})$ and $(r_0, \dot{r}_0, h, i, \theta)$. Nevertheless, in practical terms, the latter quantities can be computed using an external high-fidelity propagator. This can be representative of an on-board absolute state estimator. Alternatively, these quantities can be included in the integration step of the dynamical model using an approximated absolute dynamical model. An acceptable set of differential equations for the reference orbit is suggested here:

$$\ddot{r}_0 = -\frac{\mu}{r_0^2} + \frac{h^2}{r_0^3} - \frac{k_{J2}}{r_0^4} (1 - 3s_i^2 s_\theta^2)$$

$$\dot{h} = -\frac{k_{J2}s_i^2 s_{2\theta}}{r_0^3}$$

$$\dot{\Omega} = -\frac{2k_{J2}s_\theta^2 c_i}{hr_0^3}$$

$$\frac{d}{dt}i = -\frac{k_{J2}s_{2i}s_{2\theta}}{2hr_0^3}$$

$$\dot{\theta} = \frac{h}{r_0^2} + \frac{2k_{J2}c_i^2 s_\theta^2}{hr_0^3}$$

Relative dynamics modeling using relative orbital elements

Up to this point, we have parametrized the motion of the spacecraft in the most natural manner: using position and velocity expressed in a given reference frame. An important and rapidly increasing approach to describe the

relative motion is to use a combination of the relevant orbital elements of the reference and spacecraft orbits.

Indeed, we developed our models assuming there exists a reference frame that moves along a reference orbit and a spacecraft nearby, which is flying its own orbit, although quite close to the reference one. In general, in the 2BP, we can parametrize the orbits using the Keplerian elements. Even though the following reasoning can be developed using the classical set of orbital elements, we use a slightly different parametrization that will help us, namely:

$$\chi = \begin{pmatrix} a \\ u \\ e_x \\ e_y \\ i \\ \Omega \end{pmatrix} = \begin{pmatrix} a \\ \omega + M_0 \\ e \cos(\omega) \\ e \sin(\omega) \\ i \\ \Omega \end{pmatrix}$$

where M is the mean anomaly, a the semimajor axis, e the eccentricity, i the orbit inclination, ω the argument of perigee, and Ω the RAAN. The classical set of Keplerian elements is modified considering the eccentricity vector $e = (e_x, e_y)^T$ and the mean argument of longitude u to avoid singularities for near-circular orbits. If we think at two orbits close to each other, we can easily, and quite intuitively, introduce a set of relative orbital elements $\delta\chi$ that is a combination of the orbital elements of the spacecraft and reference orbit. There exist multiple ways one could define the set of relative orbital elements: one could use the pure mathematical difference $\Delta\chi = \chi_s - \chi_r$; nevertheless, researchers have tried to define smart combination of orbital elements to derive a set of relative orbital elements, which could be robust to typical singularities that may be encountered in orbit representation [35–38].

One common set of relative orbital elements is the one proposed by D'Amico and later extended [35,37]. The set of quasisingular relative orbital elements reads:

$$\delta\chi = \begin{pmatrix} \delta a \\ \delta\lambda \\ \delta e_x \\ \delta e_y \\ \delta i_x \\ \delta i_y \end{pmatrix} = \begin{pmatrix} (a_s - a_r)/a_r \\ (u_s - u_r) + (\Omega_s - \Omega_r)\cos(i) \\ e_{xs} - e_{xr} \\ e_{ys} - e_{yr} \\ i_s - i_r \\ (\Omega_s - \Omega_r)\sin(i) \end{pmatrix} \quad (4.97a)$$

where the s subscript refers to the spacecraft, r to the reference orbit. The semimajor axis difference has been normalized through the chief semimajor axis to have dimensionless quantities. $\delta\lambda$ denotes the relative mean longitude between the spacecraft. Apart from δa and $\delta\lambda$, the relative orbit parameterization is based on the relative eccentricity and inclination vectors for which the following Cartesian and polar notations are applied:

$$\begin{aligned}\delta e &= \begin{pmatrix} \delta e_x \\ \delta e_y \end{pmatrix} = \boldsymbol{\delta e} \begin{pmatrix} \cos(\phi) \\ \sin(\phi) \end{pmatrix} \\ \delta i &= \begin{pmatrix} \delta i_x \\ \delta i_y \end{pmatrix} = \boldsymbol{\delta i} \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}\end{aligned}\tag{4.97b}$$

The amplitudes (or lengths) of the relative e/i -vectors are denoted by δe and δi , respectively, and should not be confused with the arithmetic differences of eccentricity and inclination. The phases of the relative e/i -vectors are termed relative perigee ϕ and relative ascending node θ because they characterize the geometry of the relative orbit as seen by the chief spacecraft. In particular, as will be shown later, ϕ and θ determine the angular locations of the perigee and ascending node of the relative orbit.

The benefit of using such model is that, if the perturbations are neglected, the geometry of the relative motion with respect to a reference orbit is uniquely determined by a set of invariant relative orbital elements, except for the relative mean longitude, which follows the Keplerian propagation. Indeed, in absolute terms, we the natural evolution of the dynamic system can be described only by the rate of change of the mean anomaly:

$$\begin{aligned}\frac{dM_0}{dt} &= \sqrt{\frac{\mu}{a^3}} \text{ if } \Delta M, \Delta a \ll 1 \text{ then } \Delta \dot{M}_0 = \frac{d(\Delta M)}{dt} = -\frac{3}{2} \sqrt{\frac{\mu}{a^5}} \Delta a \\ &= -\frac{3}{2} n \frac{\Delta a}{a} = -\frac{3}{2} n \delta a\end{aligned}$$

In which only the assumption of small discrepancies between the along-track positions and orbital semimajor axes is used. It is important to remark that no constraints have been put to the relative eccentricity. The unperturbed model is then simply written as:

$$\delta \dot{\chi} = \begin{bmatrix} 0 \\ -1.5n & \mathbf{0}_{6 \times 5} \\ \mathbf{0}_{4 \times 1} \end{bmatrix}\tag{4.98}$$

Coordinates transformation

The active collision avoidance maneuvers depend on the relative metric distance between two agents. The relative distance is naturally expressed in the Cartesian LVLH reference frame. The mapping between the cartesian relative state to the Relative Orbital Elements (ROE) $\delta\chi$ is required to process the measurements and compute the guidance and control output. The transformation matrices are derived by using the classical orbital elements difference $\Delta OE = [\Delta a \Delta M \Delta\omega \Delta e \Delta i \Delta\Omega]$ as follows:

$$J_{\delta\chi}^X = \frac{\partial X}{\partial \Delta OE} \cdot \frac{\partial \Delta OE}{\partial \delta\chi}, J_{\delta\chi}^{\delta\chi} = \frac{\partial \delta\chi}{\partial \Delta OE} \cdot \frac{\partial \Delta OE}{\partial X}$$

where a is the semimajor axis, M_θ is the mean anomaly, ω the argument of perigee, e the eccentricity, i the inclination, and Ω the RAAN. The first-order approximation of the mapping between the Hill state and classical osculating orbital elements yields:

$$\begin{aligned} \delta x &= \frac{r_0}{a} \Delta a - a \cdot \cos \theta \Delta e + \frac{ae \sin \theta}{\sqrt{1-e^2}} \Delta M \\ \delta y &= \left(a + \frac{r_0}{1-e^2} \right) \sin \theta \Delta e + \frac{a^2}{r_0} \sqrt{1-e^2} \Delta M + r_0 \Delta \omega + r_0 \cos i \Delta \Omega \\ \delta z &= r_0 \sin(\theta + \omega) \Delta i - r_0 \sin i \cos(\theta + \omega) \Delta \Omega \end{aligned} \tag{4.99}$$

By differentiation, the full transformation is obtained:

$$\begin{aligned} \delta \dot{x} &= -\frac{ne \sin \theta}{2\sqrt{1-e^2}} \Delta a + n \sin \theta \sqrt{1-e^2} \left(\frac{a^3}{r_0^2} \right) \Delta e + en \cos \theta \frac{a^3}{r_0^2} \Delta M \\ \delta \dot{y} &= \left[n \sqrt{1-e^2} \left(1 + \frac{r_0}{a(1-e^2)} \right) \left(\frac{a^3}{r_0^2} \right) \cos \theta + \frac{aen \sin^2 \theta}{(1-e^2)^{\frac{3}{2}}} \right] \Delta e \\ &\quad - en \sin \theta \frac{a^3}{r_0^2} \Delta M + \frac{aen \sin \theta}{\sqrt{1-e^2}} \Delta \omega \\ \delta \dot{z} &= \frac{an}{\sqrt{1-e^2}} (\sin i [\sin(\theta + \omega) + e \sin \omega] \Delta \Omega + [\cos \theta + \omega + e \cos \omega] \Delta i) \end{aligned}$$

Combining the equations, the transformation matrix between Hill state and classical orbital elements ΔOE , namely $\frac{\partial X}{\partial \Delta OE}$ and its inverse, can be obtained. To formulate the complete transformation, the Jacobian of the

transformation between classical orbital elements and relative orbital elements $\delta\chi$ is required. Such transformation is obtained from the definition of $\delta\chi$ for $\Delta OE \rightarrow 0$:

$$\frac{\partial \Delta OE}{\partial \delta\chi} = \begin{bmatrix} a & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \frac{\sin(\omega)}{e} & -\frac{\cos(\omega)}{e} & 0 & \frac{\cos(i)}{\sin(i)} \\ 0 & 0 & -\frac{\sin(\omega)}{e} & \frac{\cos(\omega)}{e} & 0 & 0 \\ 0 & 0 & \cos(\omega) & \sin(\omega) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sin(i) \end{bmatrix}$$

$$\frac{\partial \delta\chi}{\partial \Delta OE} = \begin{bmatrix} \frac{1}{a} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & \cos(i) \\ 0 & 0 & -e \sin(\omega) & e \cos(\omega) & 0 & 0 \\ 0 & 0 & e \cos(\omega) & \sin(\omega) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sin(i) \end{bmatrix}$$

The described model is very useful to perform the design of the relative trajectories. Indeed, such model parametrization is very much used in formation flying.

At this point, we need to make an important consideration to clear up the potential ambiguity the reader may find while choosing between different models, or better, model parametrization. If the first-order mapping between the Cartesian and ROE state in Eq. (4.99), which is valid for any eccentricity, is furtherly constrained with the additional assumption of nearly circular reference orbit, one would find [35]:

$$\begin{aligned} \delta x/a &= \delta a - \delta e_x \cdot \cos u - \delta e_y \cdot \sin u \\ \delta y/a &= -\frac{3}{2} \delta au + \delta \lambda + 2\delta e_x \cdot \cos u - 2\delta e_y \cdot \sin u \\ \delta z/a &= \delta i_x \cdot \sin u - \delta i_y \cdot \cos u \end{aligned} \tag{4.100}$$

where u is the mean argument of latitude. The equations directly match the parametrization given in [Eq. \(4.96a and b\)](#). Indeed, the relative orbital elements can be thought as the integration constants of the C–W equations. The geometrical meaning is then trivially derived. For instance, we can already expect that the term $-\frac{3}{2}\delta a + \delta\lambda$ replaces the combination of C_1 and C_2 ; hence, we could expect that δa and $\delta\lambda$ will be responsible for the ellipse center shift and drift.

To better highlight the abovementioned considerations, we can use a slightly modified version of [Eq. \(4.100\)](#). If we rearrange [Eq. \(4.100\)](#) using the polar representation of $\delta\mathbf{e}$ and $\delta\mathbf{i}$ vectors in [Eq. \(4.97b\)](#), we obtain:

$$\begin{aligned}\delta x/a &= \delta a - \delta e \cdot \cos(u - \Phi) \\ \delta y/a &= -\frac{3}{2}\delta au + \delta\lambda + 2\delta e \cdot \sin(u - \Phi) \\ \delta z/a &= \delta i \cdot \sin(u - \theta)\end{aligned}\tag{4.101}$$

Relative motion geometry

The mapping between ROE state and Cartesian–Hill state has been described in the previous section. Hence, the reader should already be familiar with the geometry of the resultant relative motion. As already stated, and important features of ROE parametrization are the unperturbed motion is characterized by a set of five invariant relative orbital elements and a free coordinate that indicates the relative phase angle along the relative orbit.

Analyzing [Eq. \(4.101\)](#), we can derive the necessary conditions for bounded, centered relative motion of a spacecraft with respect to the moving reference frame (Δ or Γ):

$$\delta a = 0$$

$$\delta\lambda = 0 \leftrightarrow \Delta u = -\Delta\Omega\cos(i)$$

where the definition of the relative orbital elements has been used. If these conditions apply, then we obtain an elliptical motion as in [Fig. 4.11](#), whose semimajor axis lies on the along-track direction and equals to $2a\delta e$. The semiminor axis instead lies on the radial direction and equals to $a\delta e$. The in-plane motion geometry is fully defined by the relative eccentricity vector. The size of the ellipse is defined by the absolute value $\delta e = |\delta\mathbf{e}|$, whereas the phase angle of the polar representation Φ represents the relative pericenter. When the mean argument of latitude u equals Φ , the spacecraft is below the origin along the radial direction. Similarly, the relative inclination

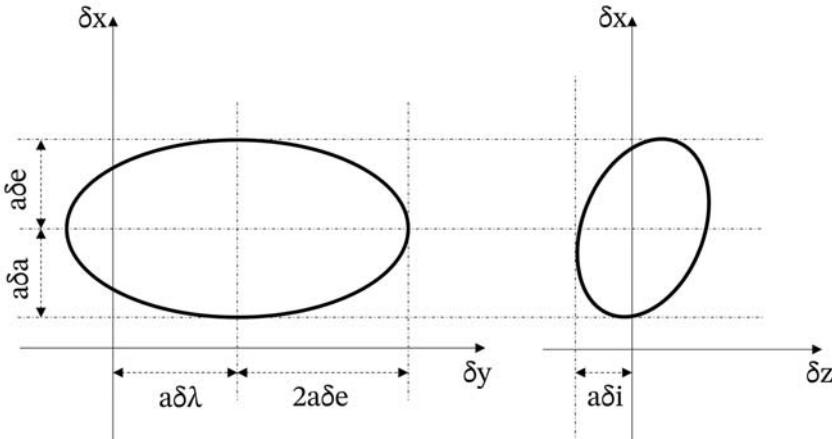


Figure 4.11 Relative motion geometry parametrized using relative orbital elements.

vector defines the cross-track motion, which is a harmonic oscillation with amplitude $a\delta i$ and phase angle $u - \theta$.

The drifting trajectories are natural drift orbits achieved by a different semimajor axis. The different semimajor axis results in slightly different periods, which determines the relative drift velocity. These trajectories allow helicoidal trajectories enclosing the along-track direction. Given a drift distance (d_{drift}) and duration (T_{drift}), we can determine the drift rate and consequently the $a\delta a$ necessary to enter the ballistic drift by inverting the dynamics equation expressed in Eq. (4.98).

$$\dot{\lambda} = -\frac{3}{2} n \delta a \rightarrow a\delta a = -\frac{2d_{\text{drift}}}{3T_{\text{drift}}n}$$

where n is the mean motion.

Energy-matching condition and passive safety

The parametrization using relative orbital elements allows deriving straightforward relationship to achieve motion characteristics, such as bounded relative orbits and passive safety.

To avoid the relative drift, it is critical that the relative motion of the spacecrafts remains bounded. A fundamental concept in spacecraft relative motion is the orbital energy-matching method to generate bounded formations. The orbital energy of the satellites is a function of the semimajor axis only:

$$\mathcal{E} = -\frac{\mu}{2a}$$

Hence, it is sufficient to match the orbital energy of the reference orbit to generate bounded formations. In order to work out relevant initial conditions, being either in the Cartesian space or $\delta\chi$, we refer to the $\delta\chi$ relative space. It is sufficient that $\delta a = 0$ for the relative orbital elements defining the relative motion. Fig. 4.12 shows trajectories propagation based on energy-matching initial conditions in perturbed models.

A remarkable observation on passively safe trajectories can be derived from the relationship between $\delta\mathbf{e}$ and $\delta\mathbf{i}$ vectors. The idea of using e/i-vector separation to collocate satellites has been studied for many years in the context of geostationary satellites station-keeping. A similar approach can be used for formation flying and in general relative trajectories design.

In such application, the presence of along-track location uncertainties leads to the criticality of designing a relative trajectory that can properly separate the satellites in the radial and cross-track direction.

If we take Eq. (4.101) and calculate the minimum relative distance as a function of the mean argument of latitude, one can write:

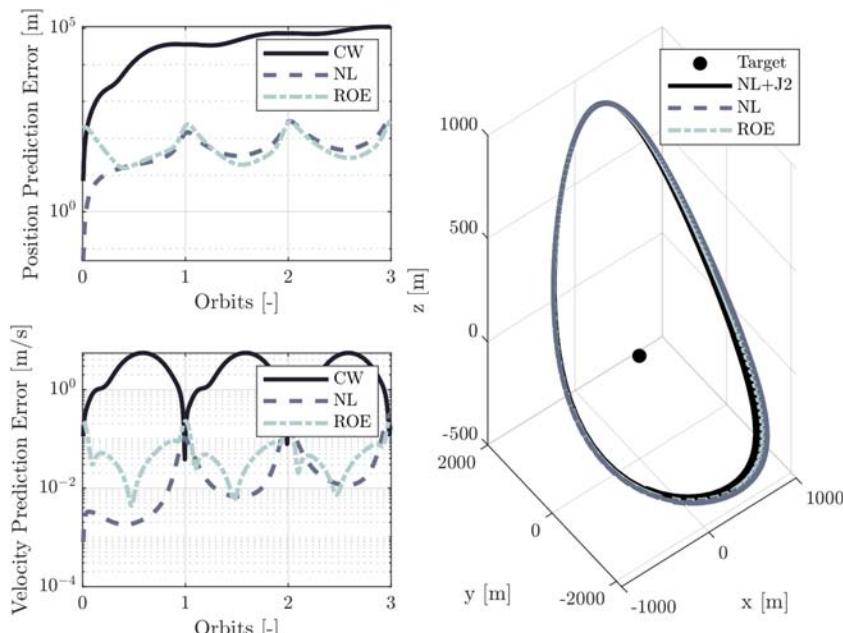


Figure 4.12 Energy-matching condition and bounded orbits for highly eccentric reference orbits [39].

$$d_{xz} = \frac{\sqrt{2}|\delta\mathbf{e} \cdot \delta\mathbf{i}|}{(\delta e^2 + \delta i^2 + |\delta\mathbf{e} + \delta\mathbf{i}| \cdot |\delta\mathbf{e} - \delta\mathbf{i}|)^{1/2}}$$

The expression shows that the minimum distance projected on the xz plane is maximized when the $\delta\mathbf{e}$ and $\delta\mathbf{i}$ vectors are either parallel or antiparallel. This condition ensures that, when the spacecraft crosses the orbital plane (cross-track distance vanished), it is at its maximum radial distance and vice versa. In contrast, if we take the condition $\delta\mathbf{e} \perp \delta\mathbf{i}$, the radial and cross-track distances can vanish simultaneously, yielding a trajectory that intersect the along-track axis.

Perturbed relative dynamics with relative orbital elements

Similar to the Cartesian models, researchers have expanded the relative dynamical model parametrized with relative orbital elements to a J_2 -perturbed dynamics. Here we report only the formulation: for a detailed derivation, the reader is suggested to refer to Ref. [37]. The complete dynamical model can be expressed as:

$$\dot{\delta\chi} = (\mathbf{A}_k + \mathbf{A}_{J2}) \cdot \delta\chi + \mathbf{B}u$$

$$\mathbf{A}_{J2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{7}{2}(1+\eta)(3\cos^2 i_r - 1) & 0 & e_x GFP & e_y GFP & -FS & 0 \\ \frac{7}{2}e_y Q & 0 & -4e_x e_y GQ & -(1+4G e_y^2)Q & 5e_y S & 0 \\ -\frac{7}{2}e_x Q & 0 & (1+4G e_x^2)Q & 4e_x e_y GQ & -5e_x S & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{7}{2}S & 0 & -4e_x GS & -4e_y GS & 2T & 0 \end{bmatrix}$$

where the r subscript refers to the reference orbits and the contributing terms are:

$$\begin{aligned} k &= \gamma a_r^{-\frac{7}{2}} \eta^{-4}, \quad \eta = \sqrt{1 - e_r^2}, \quad \gamma = \frac{3}{4} J_2 R_e^2 \sqrt{\mu}, \quad e_x = e_r \cos \omega_r, \quad e_y \\ &= e_r \sin \omega_r, \quad E = 1 + \eta, \quad G = \frac{1}{\eta^2}, \end{aligned}$$

$$F = 4 + 3\eta, \quad P = 3 \cos^2 i_r - 1, \quad Q = 5 \cos^2 i_r - 1, \quad S = \sin 2i_r, \quad T = \sin^2 i_r$$

The control matrix is derived from Gauss Variational Equation as in Ref. [37]:

$$B = \frac{1}{an} \begin{bmatrix} \frac{2}{\eta} e_r \sin(\theta) & \frac{2}{\eta} (1 + e_r \cos(\theta)) & 0 \\ -\frac{2\eta^2}{(1 + e_r \cos(\theta))} & 0 & 0 \\ \eta \sin(\omega_r) + \theta & \eta \frac{(2 + e_r \cos(\theta)) \cos(\omega_r + \theta) + e_x}{1 + e_r \cos(\theta)} & \eta \frac{e_y}{\tan(i_r)} \frac{\sin(\omega_r + \theta)}{1 + e_r \cos(\theta)} \\ -\eta \cos(\omega_r) + \theta & \eta \frac{(2 + e_r \cos(\theta)) \sin(\omega_r + \theta) + e_y}{1 + e_r \cos(\theta)} & -\eta \frac{e_x}{\tan(i_r)} \frac{\sin(\omega_r + \theta)}{1 + e_r \cos(\theta)} \\ 0 & 0 & \eta \frac{\cos(\omega_r + \theta)}{1 + e_r \cos(\theta)} \\ 0 & 0 & \eta \frac{\sin(\omega_r + \theta)}{1 + e_r \cos(\theta)} \end{bmatrix}$$

A very useful visualization of the effects of Earth oblateness on relative motion is given by Koenig in Ref. [37]. If we plot the combined effects of J_2 and Keplerian motion of the evolution of the relative orbital elements, we could produce Fig. 4.13.

A modal decomposition of the combined effects of Keplerian relative motion and J_2 is illustrated in Fig. 4.13. The dotted lines denote individual modes, and solid lines denote combined trajectories. Each of these plots superimposes the motion of each of three state component pairs. The first pair includes the relative semimajor axis and mean along-track separation, the second pair includes state components that are functions of the eccentricity and argument of perigee, and the third pair includes components that are functions of the inclination and Ω . The combined effects of Keplerian relative motion and J_2 produce four distinct relative motion modes:

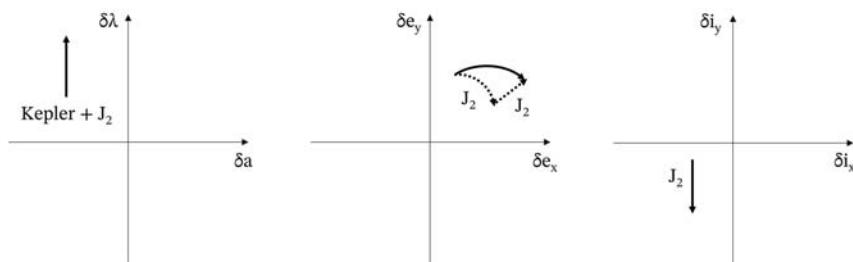


Figure 4.13 J_2 effects on the evolution of the relative orbital elements.

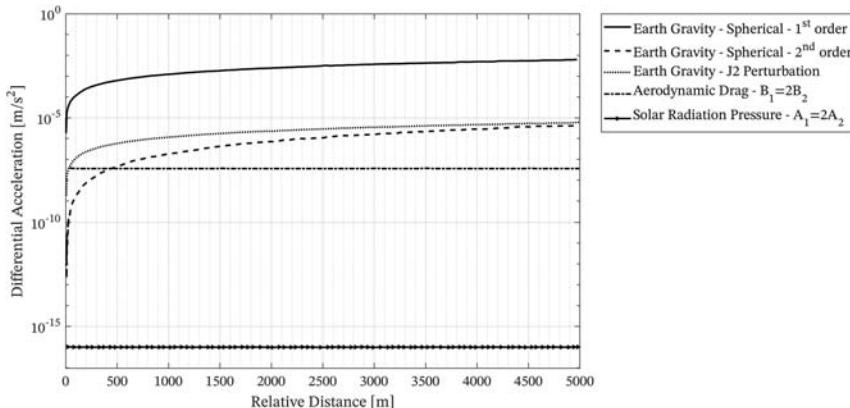


Figure 4.14 Magnitude of relative accelerations for close nearly circular motion as a function of the relative distances.

- a constant drift of $\delta\lambda$ due to both Keplerian relative motion and J_2
- a rotation of the relative eccentricity vector due to J_2
- a secular drift of the relative eccentricity vector proportional to the chief eccentricity and orthogonal to the phase angle of the chief argument of perigee due to J_2
- a constant drift of δi_y due to J_2 .

Beside Earth oblateness, the orbital perturbations arise from multiple sources as discussed in [Chapter 3](#) – The Space environment. The magnitude of relative accelerations for close near-circular relative motion in LEOs (< 1500 km) as a function of the relative distance is shown in [Fig. 4.14](#). For a more complete analysis, please refer to Ref. [35].

Comparison of relative dynamics modeling

A thorough comparison between different relative dynamical models is beyond the scope of this book. Nevertheless, we can take advantage of literature results reporting comprehensive and exhaustive results on the use of different analytical models [40]. We report here the comparison between the following models (some of them have been fully described in the chapter):

- *Clohessy–Wiltshire model, or Hill–Clohessy–Wiltshire model (C–W, CW, HCV)*. Keplerian nearly circular reference orbits based on Cartesian translational states [1].
- *Quadratic Volterra (QV) model*. Keplerian nearly circular reference orbits based on Cartesian translational states [41].

- *Schweighart–Sedwick model.* Perturbed nearly circular reference orbits based on Cartesian translational states [42].
- *Yamanaka–Ankersen STM.* Keplerian eccentric reference orbits based on Cartesian translational states [33].
- *Broucke STM.* Keplerian eccentric reference orbits based on Cartesian translational states [43].
- *GAM STM.* Perturbed nearly circular reference orbits based on orbital element states [44].
- *Gim–Alfriend STM.* Perturbed eccentric reference orbits based on orbital element states [36].
- *KGD STM.* Perturbed eccentric reference orbits based on orbital element states [37].
- *Yan–Alfriend nonlinear theory.* Perturbed eccentric reference orbits based on orbital element states [45].
- *Biria–Russel–Vinti method.* Perturbed eccentric reference orbits [46].

The comparison between the models is performed for a 24 h simulation for each test case. The metric is the error in position prediction. The benchmark is represented by a high-fidelity propagator, which entails the forces contribution of 120×120 gravity field, atmospheric drag, solar radiation pressure, third-body sun and moon, relativistic and tidal effects. Tables 4.4–4.6 report propagation results for reference scenarios.

Table 4.4 Scenario: fixed relative motion geometry with small separation (~ 300 m) and a chief orbit with eccentricity varied over the range 10^{-4} to 0.7.

Prediction Errors [m]—Order of magnitude

Model	Eccentricity [-]				
	10^{-4}	10^{-3}	10^{-2}	10^{-1}	0.7
CW/QV	$3.5 \cdot 10^2$	$4.0 \cdot 10^2$	$1.0 \cdot 10^3$	$9.0 \cdot 10^3$	$4.5 \cdot 10^4$
Yamanaka–Ankersen/ Broucke	$3.0 \cdot 10^2$	$3.0 \cdot 10^2$	$5.0 \cdot 10^2$	$7.0 \cdot 10^2$	$1.5 \cdot 10^3$
Schweighart–Sedwick	$2.5 \cdot 10^2$	$2.5 \cdot 10^2$	$5.0 \cdot 10^2$	$1.5 \cdot 10^3$	$6.0 \cdot 10^3$
Gim–Alfriend	$2.0 \cdot 10^2$	$2.0 \cdot 10^2$	$1.5 \cdot 10^2$	$5.0 \cdot 10^1$	$5.0 \cdot 10^1$
Yan–Alfriend/KGD	$2.0 \cdot 10^2$	$2.0 \cdot 10^2$	$1.5 \cdot 10^2$	$5.0 \cdot 10^1$	$5.0 \cdot 10^1$
GAM	$2.0 \cdot 10^2$	$2.0 \cdot 10^2$	$1.5 \cdot 10^2$	$7.0 \cdot 10^1$	$1.5 \cdot 10^2$
Biria–Russel–Vinti	$2.0 \cdot 10^2$	$2.0 \cdot 10^2$	$1.5 \cdot 10^2$	$5.5 \cdot 10^1$	$5.0 \cdot 10^1$

Inspired by J. Sullivan, S. Grimberg, S. D’Amico, Comprehensive survey and assessment of spacecraft relative motion dynamics models, Journal of Guidance, Control, and Dynamics 40 (8) (2017) 1837–1859. <https://doi.org/10.2514/1.G002309>.

Table 4.5 Scenario: nearly circular sun-synchronous LEO with perigee altitude of 750 km and a nominally bounded a centered relative motion trajectory. The relative distance is varied from approximately 2 m to 250 km.

Prediction Errors [m]—Order of magnitude

Model	$\delta r_{\max}/a [-]$				
	$3 \cdot 10^{-5}$	10^{-4}	10^{-3}	10^{-2}	$6.5 \cdot 10^{-2}$
CW	$2.0 \cdot 10^2$	$4.0 \cdot 10^2$	$7.0 \cdot 10^3$	$1.5 \cdot 10^5$	$3.0 \cdot 10^6$
QV	$2.0 \cdot 10^2$	$4.0 \cdot 10^2$	$6.0 \cdot 10^3$	$9.0 \cdot 10^4$	$8.0 \cdot 10^5$
Yamanaka–Ankersen/ Brouke	$2.0 \cdot 10^2$	$2.5 \cdot 10^2$	$2.0 \cdot 10^3$	$2.0 \cdot 10^4$	$1.5 \cdot 10^5$
Schweighart–Sedwick	$2.0 \cdot 10^2$	$2.5 \cdot 10^2$	$1.5 \cdot 10^3$	$1.5 \cdot 10^4$	$1.5 \cdot 10^5$
Gim–Alfriend	$2.0 \cdot 10^2$	$2.0 \cdot 10^2$	$6.0 \cdot 10^2$	$7.0 \cdot 10^4$	$3.0 \cdot 10^6$
Yan–Alfriend	$2.0 \cdot 10^2$	$2.0 \cdot 10^2$	$2.2 \cdot 10^2$	$3.0 \cdot 10^2$	$4.5 \cdot 10^2$
KGD	$2.0 \cdot 10^2$	$2.0 \cdot 10^2$	$2.2 \cdot 10^2$	$3.0 \cdot 10^2$	$3.0 \cdot 10^3$
GAM	$2.0 \cdot 10^2$	$2.0 \cdot 10^2$	$2.2 \cdot 10^2$	$3.5 \cdot 10^2$	$3.0 \cdot 10^3$
Biria–Russel–Vinti	$2.0 \cdot 10^2$	$2.0 \cdot 10^2$	$2.0 \cdot 10^3$	$6.0 \cdot 10^4$	$6.0 \cdot 10^5$

Inspired by J. Sullivan, S. Grimberg, S. D’Amico, Comprehensive survey and assessment of spacecraft relative motion dynamics models, Journal of Guidance, Control, and Dynamics 40 (8) (2017) 1837–1859. <https://doi.org/10.2514/1.G002309>.

Table 4.6 Scenario: highly inclined, eccentric reference orbit with $e = 0.5$ and a perigee altitude of 750 km. The relative distance is varied from approximately 2 m to 250 km.

Prediction Errors [m]—Order of magnitude

Model	$\delta r_{\max}/a [-]$				
	$6 \cdot 10^{-6}$	10^{-4}	10^{-3}	10^{-2}	$7 \cdot 10^{-2}$
CW	$6.0 \cdot 10^1$	$4.0 \cdot 10^3$	$3.0 \cdot 10^4$	$4.0 \cdot 10^5$	$6.0 \cdot 10^6$
QV	$6.0 \cdot 10^1$	$4.0 \cdot 10^3$	$3.0 \cdot 10^4$	$3.5 \cdot 10^5$	$3.0 \cdot 10^6$
Yamanaka–Ankersen/ Brouke	$6.0 \cdot 10^1$	$3.0 \cdot 10^2$	$7.0 \cdot 10^3$	$5.0 \cdot 10^5$	$2.5 \cdot 10^6$
Schweighart–Sedwick	$6.0 \cdot 10^1$	$4.0 \cdot 10^3$	$3.0 \cdot 10^4$	$3.5 \cdot 10^5$	$1.5 \cdot 10^5$
Gim–Alfriend	$6.0 \cdot 10^1$	$1.0 \cdot 10^2$	$9.0 \cdot 10^2$	$5.0 \cdot 10^4$	$4.0 \cdot 10^6$
Yan–Alfriend	$6.0 \cdot 10^1$	$6.0 \cdot 10^1$	$5.0 \cdot 10^1$	$4.0 \cdot 10^2$	$3.0 \cdot 10^3$
KGD	$6.0 \cdot 10^1$	$6.0 \cdot 10^1$	$5.0 \cdot 10^1$	$4.0 \cdot 10^2$	$4.0 \cdot 10^3$
GAM	$6.0 \cdot 10^1$	$1.0 \cdot 10^2$	$2.5 \cdot 10^2$	$4.0 \cdot 10^3$	$3.0 \cdot 10^4$
Biria–Russel–Vinti	$6.0 \cdot 10^1$	$6.0 \cdot 10^1$	$6.5 \cdot 10^1$	$8.0 \cdot 10^2$	$2.0 \cdot 10^4$

Inspired by J. Sullivan, S. Grimberg, S. D’Amico, Comprehensive survey and assessment of spacecraft relative motion dynamics models, Journal of Guidance, Control, and Dynamics 40 (8) (2017) 1837–1859. <https://doi.org/10.2514/1.G002309>.

Cartesian and relative orbital elements mapping

The coordinates transformation has been presented in the previous section. However, some concerns may arise with respect to the accuracy of such model, reason for which a set of propagation tests have been performed, in order to assess the level of confidence to pose into the ROE formulation. The dynamics has indeed been compared to different Cartesian formulations of the relative dynamics and to a high-fidelity orbital propagator. The procedure for the validation of the ROE formulation with the high-fidelity propagator is presented in Fig. 4.15.

The selection of the initial conditions for the propagation of the Cartesian state is fundamental. An incoherence among the different methods can arise for what concerns the different level of linearization required by the translation of the ROE into the Cartesian state. Indeed, different state propagations arise if such conversion is performed through the nonlinear chain of transformations described in the bottom branch Fig. 4.12 or through the direct linear transformation, as in the last conversion of the upper branch of the graph. The former starts from the ROE and, passing through the difference in orbital elements ΔOE and the reference orbital elements OE_{ref} , recovers the inertial state which can be remapped to the relative Cartesian state in the LVLH frame. The latter uses instead a direct transformation from the initial ROE to the relative Cartesian frame, implying a higher degree of linearization. Details on the two mappings can be found in previous works [44]. Fig. 4.16 presents the position error δr accuracy obtained by a nonlinear Cartesian relative dynamics model, when initialized with the linear transformation (left plot) and the nonlinear one (right plot). The propagation of the ROE with the given formulation is also presented for comparison. The position errors δr are computed with respect to the trajectory propagated with the high-fidelity orbital simulator, over a time span of a complete day. It is easy to appreciate the much higher drift rate of the nonlinear Cartesian model exploiting the linearized initial condition, with respect to the evolution using the nonlinear transformation. On the contrary, as highlighted also in other works [44], when dealing with a linearized

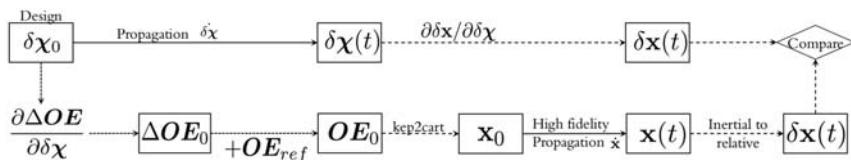


Figure 4.15 Propagation scheme for Cartesian and relative orbital elements.

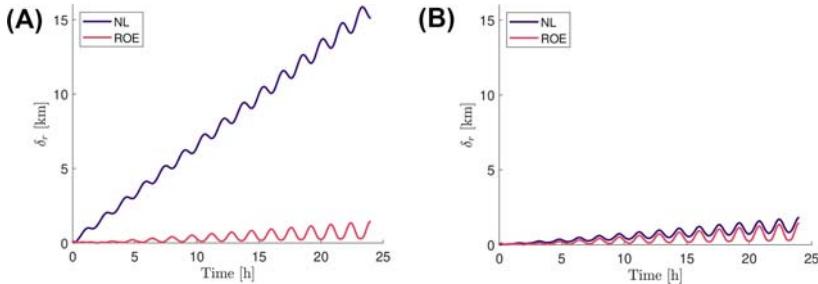


Figure 4.16 Comparison of the position error of the (A) nonlinear Cartesian relative dynamics and (B) ROE formulation with respect to the high-fidelity simulator. The initial conditions for the nonlinear model are generated using the linear transformation (left) or the nonlinear derivation.

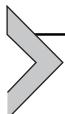
dynamics, such as the C–W equations, in order to keep the consistency in terms of nonlinear effects, the linearized transformation ensures a better representation of the system. Nevertheless, this analysis also gave the hint on the feasibility of exploiting the ROE with J_2 perturbations for the design of the relative trajectories, without introducing huge inaccuracies, as seen by the plots in Fig. 4.16, where a maximum of ~ 2 km error is introduced over a one-day propagation arc.

References

- [1] H. Curtis, Orbital Mechanics for Engineering Students, Elsevier, 2005.
- [2] D.A. Vallado, Fundamentals of Astrodynamics and Applications, vol. 12, Springer Science & Business Media, 2001.
- [3] R.H. Battin, An Introduction to the Mathematics and Methods of Astrodynamics, Aiaa, 1999.
- [4] R.R. Bate, D.D. Mueller, J.E. White, W.W. Saylor, Fundamentals of Astrodynamics, second ed., Dover Publications, New York, 2020.
- [5] V.A. Chobotov, Orbital Mechanics, American Institute of Aeronautics and Astronautics, 2002.
- [6] J.E. Prussing, B.A. Conway, Orbital Mechanics, Oxford University Press, USA, 1993.
- [7] K.B. Bhatnagar, L.M. Saha, N-body problem, Bulletin of the Astronomical Society of India 21 (1993) 1–25.
- [8] R.A. Howland, Intermediate Dynamics: A Linear Algebraic Approach, Springer Science & Business Media, 2005.
- [9] Y. Kozai, The motion of a close earth satellite, The Astronomical Journal 64 (1959) 367.
- [10] J.H. Verner, Explicit Runge–Kutta methods with estimates of the local truncation error, SIAM Journal on Numerical Analysis 15 (4) (1978) 772–790.
- [11] L.F. Shampine, M.K. Gordon, Computer Solution of Ordinary Differential Equations: The Initial Value Problem, Freeman, San Francisco, 1975.
- [12] J. Barrow-Green, Poincaré and the Three Body Problem, No. 11, American Mathematical Soc., 1997.

- [13] V. Szebehely, Theory of Orbits: The Restricted Problem of Three Bodies, Academic Press, New York and London, 1967.
- [14] F. Ferrari, Non-keplerian models for mission analysis scenarios about solar system bodies, Ph.D. Dissertation, 2016.
- [15] K.C. Howell, Three-dimensional, periodic, ‘halo’ orbits, *Celestial Mechanics* 32 (1) (1984) 53–71.
- [16] M.W. Lo, Halo orbit generation using the center manifold, *Advances in the Astronautical Sciences* 95 (part 1) (1997) 109–116.
- [17] R. Broucke, Stability of periodic orbits in the elliptic, restricted three-body problem, *AIAA Journal* 7 (6) (1969) 1003–1009.
- [18] J.D. Hadjidemetriou, Resonant motion in the restricted three body problem, *Celestial Mechanics and Dynamical Astronomy* 56 (1) (1993) 201–219.
- [19] A.M. Legendre, Recherches sur l’attraction des sphéroïdes homogènes, De l’Imprimerie Royale, 1785.
- [20] E.W. Hobson, The Theory of Spherical and Ellipsoidal Harmonics, Cambridge University Press, 1931.
- [21] S.L. Belousov, Tables of Normalized Associated Legendre Polynomials, Mathematical Tables Series, 2014.
- [22] A. Konopliv, S. Asmar, B. Bills, et al., The dawn gravity investigation at vesta and ceres, *Space Science Reviews* 163 (2011) 461–486, <https://doi.org/10.1007/s11214-011-9794-8>.
- [23] J.K. Miller, A.S. Konopliv, P.G. Antreasian, et al., Determination of shape, gravity, and rotational state of asteroid 433 eros, *Icarus* 155 (1) (2002) 3–17, <https://doi.org/10.1006/icar.2001.6753>.
- [24] A. Pasquale, S. Silvestrini, A. Capannolo, et al., Non-uniform Gravity Filed Model on Board Learning during Small Bodies Proximity Operations, International Astronautical Congress, Washington D.C., USA, 2019.
- [25] Y. Takahashi, D.J. Scheeres, Small body surface gravity fields via spherical harmonic expansions, *Celestial Mechanics and Dynamical Astronomy* 119 (2) (2014) 169–206.
- [26] W.D. MacMillan, The Theory of the Potential, 1958, pp. 6–60, <https://doi.org/10.1021/ed007p2530>.
- [27] P.T. Wittick, R.P. Russell, Mascon models for small body gravity fields, in: AAS/AIAA Astrodynamics Specialist Conference, 2017.
- [28] A. Colagrossi, F. Ferrari, M. Lavagna, et al., Dynamical evolution about asteroids with high fidelity gravity field and perturbations modeling, *Advances in the Astronautical Sciences* 156 (2015) 885–903.
- [29] R.A. Werner, D.J. Scheeres, Exterior gravitation of a polyhedron derived and compared with harmonic and mascon gravitation representations of asteroid 4769 castalia, *Celestial Mechanics and Dynamical Astronomy* 65 (1997) 313–344, <https://doi.org/10.1007/bf0053511>.
- [30] A. Colagrossi, F. Ferrari, M. Lavagna, et al., Coupled dynamics analysis around asteroids by means of accurate shape and perturbations modeling, in: Proceedings of the International Astronautical Congress, IAC 2015, vol. 7, 2015, pp. 5360–5370.
- [31] T.E. Carter, State transition matrices for terminal rendezvous studies: brief survey and new example, *Journal of Guidance, Control, and Dynamics* 21 (1) (2008) 148–155, <https://doi.org/10.2514/2.4211>.
- [32] G. Inalhan, M. Tillerson, J.P. How, Relative dynamics and control of spacecraft formations in eccentric orbits, *Journal of Guidance, Control, and Dynamics* 25 (1) (2002) 48–59, <https://doi.org/10.2514/2.4874>.
- [33] K. Yamanaka, F. Ankersen, New state transition matrix for relative motion on an arbitrary elliptical orbit, *Journal of Guidance, Control, and Dynamics* 25 (1) (2002) 60–66, <https://doi.org/10.2514/2.4875>.

- [34] D. Wang, B. Wu, E.K. Poh, Satellite Formation Flying 87 (2017), <https://doi.org/10.1007/978-981-10-2383-5>.
- [35] S. D'Amico, Autonomous Formation Flying in Low Earth Orbit, 2010. <http://www.narcis.nl/publication/RecordID/oai:tudelft.nl:uuid:a10e2d63-399d-48e5-884b-402e9a105c70>.
- [36] D.-W. Gim, K.T. Alfriend, State transition matrix of relative motion for the perturbed noncircular reference orbit, Journal of Guidance, Control, and Dynamics 26 (6) (2003) 956–971, <https://doi.org/10.2514/2.6924>.
- [37] A.W. Koenig, T. Guffanti, S. D'Amico, New state transition matrices for spacecraft relative motion in perturbed orbits, Journal of Guidance, Control, and Dynamics 40 (7) (2017) 1749–1768, <https://doi.org/10.2514/1.G002409>.
- [38] H. Schaub, S.R. Vadali, J.L. Junkins, K.T. Alfriend, Spacecraft formation flying control using mean orbit elements, Journal of the Astronautical Sciences 48 (1) (2001) 69–87, <https://doi.org/10.1007/bf03546219>.
- [39] S. Silvestrini, M. Lavagna, Neural-aided GNC reconfiguration algorithm for distributed space system: development and PIL test, Advances in Space Research 67 (5) (March 1, 2021) 1490–1505, <https://doi.org/10.1016/j.asr.2020.12.014>.
- [40] J. Sullivan, S. Grimberg, S. D'Amico, Comprehensive survey and assessment of spacecraft relative motion dynamics models, Journal of Guidance, Control, and Dynamics 40 (8) (2017) 1837–1859, <https://doi.org/10.2514/1.G002309>.
- [41] B.A. Newman, A.J. Sinclair, A. Lovell, A. Perez, Comparison of nonlinear analytical solutions for relative orbital motion, in: AIAA/AAS Astrodynamics Specialist Conference, AIAA Paper 2014-4163, 2014, <https://doi.org/10.2514/6.2014-4163>.
- [42] S.A. Schweighart, R.J. Sedwick, High-fidelity linearized J_2 model for satellite formation flight, Journal of Guidance, Control, and Dynamics 25 (6) (2002) 1073–1080, <https://doi.org/10.2514/2.4986>.
- [43] R.A. Broucke, Solution of the elliptic rendezvous problem with the time as independent variable, Journal of Guidance, Control, and Dynamics 26 (4) (2003) 615–621, <https://doi.org/10.2514/2.5089>.
- [44] G. Gaias, J.S. Ardaens, O. Montenbruck, Model of J_2 perturbed satellite relative motion with time-varying differential drag, Celestial Mechanics and Dynamical Astronomy 123 (4) (2015) 411–433, <https://doi.org/10.1007/s10569-015-9643-2>.
- [45] K. Alfriend, Nonlinear considerations in satellite formation flying, in: AIAA/AAS Astrodynamics Specialist Conference and Exhibit, AIAA Paper 2002-4741, August 2002, <https://doi.org/10.2514/6.2002-4741>.
- [46] A. Biria, R. Russell, A satellite relative motion model using J_2 and J_3 via vinti's intermediary, in: AIAA/AAS Space Flight Mechanics Conference, AAS Paper 16-537, Napa, CA, 2016.



Attitude dynamics

Aureliano Rivolta¹, Andrea Colagrossi², Vincenzo Pesce³,
Stefano Silvestrini²

¹D-Orbit, Fino Mornasco, Italy

²Politecnico di Milano, Milan, Italy

³Airbus D&S Advanced Studies, Toulouse, France

Attitude dynamics describes the rotational motion of any object flying in space. When considering the spacecraft dynamics, we have in mind first the orbits and the motion of the spacecraft around the Earth or another celestial object. However, in addition to this, we shall also consider the rotation of the spacecraft around its center of mass. Indeed, the spacecraft is an object moving in a three-dimensional space with a total of six degrees of freedom: three of them concerning the translational dynamics of the center of mass (i.e., orbital dynamics) and other three for the rotational dynamics about the center of mass (i.e., attitude dynamics).

Orbital and attitude dynamics are often studied as separate branches, since in most applications, there exist a substantial decoupling between the spacecraft rotational and translational dynamics. In fact, the orbital motion has a notably larger energy with respect to the attitude one, and the latter has a minor influence upon the spacecraft's orbit. Moreover, in the case of unperturbed natural dynamics, the two dynamics are actually decoupled, as assumed in the classical rigid body dynamics. It shall be noted that this is strictly true only for a rigid body, but it is practically applicable also to spacecraft with some degrees of flexibility, as noted in the Chapter 3 – The Space Environment. The orbit and attitude coupling is becoming consistent when the environmental perturbations are accounted in the analyses, even if the influence of the orbital dynamics on the attitude one is always more significant than the opposite. The effects of the attitude dynamics on the orbit begin to be relevant only for modern and future applications, whenever the spacecraft is very extended and flexible (i.e., large space structures, solar sails, etc.), or when there is a weakly stable orbit (i.e., orbits in the three-body problem (3BP) or irregular body environments).

This chapter discusses the fundamentals of spacecraft attitude dynamics with the rigid body assumption, since the flexibility can be then accounted

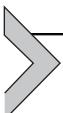
as described in the [Chapter 3](#) – The Space Environment. The content of this chapter is structured as follows:

- *Attitude kinematics* In this section, the most useful mathematical attitude representations are presented, with particular focus on the Direction Cosine Matrix (DCM), the Euler angles, the Euler axis and angle, and the quaternions. The concept of angular velocity is introduced, together with the fundamental kinematics rules for Euler angles and quaternions.
- *Attitude dynamics* This section introduces the fundamental concepts about the rigid body rotational dynamics. It includes a discussion on the inertia properties of the spacecraft, together with the derivation of the Euler equations of motion. The torque-free attitude dynamics is also presented, being the basis for any further attitude dynamics model, with some elements of attitude stability. Moreover, the motion of a spinning spacecraft is discussed and the equations of motion of a spacecraft with internal rotating elements are derived. Finally, this section describes the attitude perturbation torques, specifying the effects of the space environment on the rotational motion of a spacecraft.

The chapter is concluded with some brief sections on attitude dynamics for modern applications:

- Attitude dynamics in three-body environments.
- Relative attitude dynamics.
- Multibody spacecraft dynamics.

The reader is invited to deepen the study of spacecraft attitude dynamics in the dedicated books listed in the reference section [\[1–5\]](#).



Attitude kinematics

Attitude dynamics studies the evolution in time of the orientation of a rigid body in space under the influence of applied torques. Hence, it allows to investigate how and why the body rotates, and how fast the rotation is. However, differently from translational motion that is represented by a set of intuitive independent variables (i.e., cartesian or spherical coordinates), rotations are associated to less immediate mathematical parametrizations that are the core of attitude kinematics. Indeed, attitude dynamics is based on the attitude kinematics, which describes the rotations of a body or a system of bodies without considering the torques that cause them to move. It is the analogous of translational kinematics, but the position vector, \mathbf{r} , is substituted by one attitude parametrization describing the body rotation, and the velocity, \mathbf{v} , is replaced by the angular velocity, $\boldsymbol{\omega}$.

The mathematical parametrizations of rotations can take different forms, and the main ones will be discussed in this section, but they are all intrinsically associated to the same physical concept: the relative orientation of a reference frame with respect to another one. Thus, the first step is recalling the definition of reference frames from [Chapter 2 – Reference Systems and Planetary Models](#), considering that the most common spacecraft attitude definition relates the satellite body coordinate system, $b_1 b_2 b_3$, to an inertial (or pseudoinertial) one, such as the geocentric equatorial reference frame, IJK . In more mathematical terms, we can define a reference frame in a N dimensional space as a set of N vectors that can be used to express any point in that space. It is possible to perform transformations and rotations on such frames to change the viewpoint from which we are analyzing the system. Intuitively, whenever we look at the people sitting on a moving bus from the roadside, we see them moving in one direction, while they see us moving in the opposite direction. The same event is thus described in two different reference frames, and the same translation is described with opposite signs because we have performed a 180 degrees rotation of two people looking at each other. If we continue to follow the bus moving our head, we are performing a further rotation between the two reference frames, since we are changing the direction of our sight, and we are orienting our heads differently from our body. In this case, the head orientation can be related to the orientation of our body, and the time evolution of our line of sight can be parametrized applying a rotation to the reference system based on our body. We can generalize the concept and define attitude as the orientation of a reference system with respect to another reference system. Without reference systems, attitude loses meaning, and without attitude, two reference systems cannot share information.

We perceive reality as three-dimensional; hence, the most common reference frames for satellites are taken as three-dimensional with $N = 3$. When we talk about spacecraft attitude, the reference systems are always orthogonal, cartesian, and three-dimensional, although in some examples, a two-dimensional system can be used for simplicity. This implies that we are basing our discussion on the 3D rotation group, often denoted as $\text{SO}(3)$: the orthogonal group of dimension three. This is the group of all the rotations about the origin of three-dimensional Euclidean spaces, under the operation of composition of successive rotations. By definition, a rotation about the origin is a transformation that preserves the origin, the Euclidean distance, and the orthogonality.

Direction cosine matrix

The first and fundamental attitude parametrization we encounter in attitude kinematics is the DCM, or attitude matrix, \mathbf{A} . The DCM is the fundamental representation of a rotation. In fact, the SO(3) group theory defines the rotations as linear transformations, which can be represented by a matrix once a basis of the three-dimensional Euclidean space is defined. Sticking with the reference frames useful to satellite attitude dynamics, we will limit to orthogonal three-dimensional cartesian spaces. Hence the axes of a reference frame are three unit vectors orthogonal to each other, defining an orthonormal basis. This directly implies that the DCM has some interesting properties:

- Orthogonal 3×3 .
- Symmetric.
- Positive definite.
- Unitary determinant.

The DCMs are also known as “special orthogonal matrices,” explaining the notation SO(3).

The DCM allows us to connect two distinct reference frames to manipulate vectors expressed in different reference frames, and it is the basis to understand the attitude kinematics. Formally, the DCM contains the projections of the axis of one target reference frame (i.e., the spacecraft body frame) onto the axis of another reference frame (i.e., the inertial one):

$$\mathbf{A} = \begin{bmatrix} b_{1I} & b_{1J} & b_{1K} \\ b_{2I} & b_{2J} & b_{2K} \\ b_{3I} & b_{3J} & b_{3K} \end{bmatrix},$$

where each row represents one axis of the target reference frame.

Note that a generic 3×3 matrix contains nine elements, since three generic 3D vectors have nine independent components. However, if these vectors are unitary, the number of independent components drops to six. Then, if we state that those vectors are mutually perpendicular, with null projection, we add three constraints that drop the independent components to only three. Since the basis of a reference frame is orthonormal, the DCM belongs to this special condition, and only three components are independent and necessary to fully express a DCM.

Given a generic vector in the inertial reference frame, \mathbf{v}_{IJK} , we have three projections along each unit vector IJK , and we can express the vector through those projections. The common values that we give vectors are

simply the projections along the reference axes, and they become the way we represent and understand such vectors. A generic vector rotation from the reference IJK into reference $b_1b_2b_3$ is expressed as:

$$\mathbf{v}_b = {}_b\mathbf{A}_{IJK} \mathbf{v}_{IJK},$$

where \mathbf{v}_b is the vector expressed in the spacecraft body frame $b_1b_2b_3$.

Any rotation can be reversed easily and without additional computations. Formally we can write the rotation of the vector \mathbf{v}_b from the reference $b_1b_2b_3$ to reference frame IJK through the inverse of the DCM. In fact, mathematically speaking, we should left-multiply by the inverse to obtain the inverse relation. However, due to the orthonormality property, the DCM inverse is equal to the transpose of the DCM.

$$\mathbf{v}_{IJK} = {}_{IJK}\mathbf{A}_b \mathbf{v}_b = {}_b\mathbf{A}_{IJK}^{-1} \mathbf{v}_b = {}_b\mathbf{A}_{IJK}^T \mathbf{v}_b$$

Knowing the DCM allows us to know both direct and inverse transformations.

Note that the previous discussion can be generalized to any rotation between two generic reference frames. Moreover, we can simplify the notation for the following discussion with:

$$\mathbf{A} = {}_b\mathbf{A}_{IJK}$$

$$\mathbf{A}^T = \mathbf{A}^{-1} = {}_{IJK}\mathbf{A}_b,$$

reminding that $\mathbf{A}\mathbf{A}^T = \mathbf{I}_3$, where \mathbf{I}_3 is the 3×3 identity matrix.

The composition of successive rotations can be defined with the multiplication of the associated DCMs, resulting in another DCM. In fact, composing two rotations results again in a rotation, and the overall DCM is the product of the two individual successive rotations, taken in the reversed order compared to the sequence of rotations.

For example, assuming that we want to define a successive rotation from the body frame to a frame aligned with the spacecraft's payload, $b'_1b'_2b'_3$ as:

$$\mathbf{v}_{b'} = \mathbf{A}' \mathbf{v}_b,$$

where $\mathbf{v}_{b'}$ is a vector expressed in the payload reference frame. Then, the rotation of a vector from the inertial frame, IJK , to the payload frame becomes:

$$\mathbf{v}_{b'} = \mathbf{A}' \mathbf{v}_b = \mathbf{A}'(\mathbf{A} \mathbf{v}_{IJK}) = \mathbf{A}'\mathbf{A} \mathbf{v}_{IJK} = \mathbf{A}'' \mathbf{v}_{IJK},$$

with $\mathbf{A}'' = \mathbf{A}'\mathbf{A}$, since the vector is first rotated in $b_1 b_2 b_3$ and then in $b'_1 b'_2 b'_3$, but the matrix multiplication is taken in reversed order.

It shall be remarked that any attitude matrix shall preserve the properties of a DCM. Hence, after any numerical manipulation on these matrices, it is suggested to check if the DCM properties are still valid. If this is not the case, proper mathematical correction shall be applied to remove numerical errors and enforce the DCM properties. For example, the orthonormalization of the resulting DCM may be needed after few mathematical operations are executed on the attitude matrix.

The time evolution of the DCM is directly the time evolution of the spacecraft attitude. In fact, when the attitude matrix is known at different instants of time, it is possible to understand how the spacecraft body frame is rotating with respect to the inertial reference frame. Furthermore, composing other successive rotations, it is possible to know the rotations of any satellite-based reference frame with respect to another generic base reference frame.

Direction cosine matrices are the fundamental attitude representation parameters, and they are necessary to rotate a vector from a reference frame to a different one. However, they have nine components and few mathematical constraints making them not very suitable for guidance, navigation, and control (GNC) applications. In fact, functions used in attitude GNC usually employ a representation of the attitude having fewer parameters and fewer constraints than the attitude matrix. The next paragraphs describe the main and most used attitude parametrization methods:

- Euler angles.
- Euler axis and angle.
- Quaternions.

Euler angles

The description of the spacecraft attitude in a reference frame requires only three independent components. This leads to one of the simplest and most intuitive way to address attitude parametrization: a sequence of three rotations around three nonconsecutive axes. Indeed, it is always possible to make two orthogonal frames overlap by appropriate rotation of one of them three times around its reference axes. Each of these successive rotations is identified by an axis (commonly x , y , and z) and an angle. The set of the three rotation angles is commonly referred to as *Euler angles*. The sequence of rotations is important, and we can identify 12 different sets of three rotations to express the DCM:

- Proper Euler angles: $z-x-z$, $x-y-x$, $y-z-y$, $z-y-z$, $x-z-x$, $y-x-y$.
- Tait–Bryan, Cardan, or yaw–pitch–roll angles: $x-y-z$, $y-z-x$, $z-x-y$, $x-z-y$, $z-y-x$, $y-x-z$.

Euler angles are by far the most intuitive means of representing the attitude of a satellite, and they have a clear physical meaning. However, they suffer from computational issues.

Let us take a simple object like a six-faced dice. We try to understand the principle of Euler angles using three 90 degrees positive rotations. We define the x axis to be the normal to face 1, the y axis of face 3, and z axis of face 2, as illustrated in Fig. 5.1. The first rotation is performed around the x axis, now we have face 3 instead of face 2, and face 5 instead of face 3. Let the second rotation be around face 3 (i.e., y axis). Now we have face 1 onto original face 3 position, and face 2 where it was face 1. Last, a rotation is performed around face 2 (i.e., z axis), and we find face 4 where it was face 3 at the beginning, face 1 where it was face 2, and face 2 on top where face 1 was. The associated sequence of rotations is $x-y-z$, and the way in which these rotations have been defined is the intrinsic one. Intrinsic rotations are defined around the axes of the rotating coordinate system xyz , fixed with the moving body, which changes its orientation after each elemental rotation.

The DCM that relates the final configuration with respect to the first is:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

The procedure of generating the DCM starting from the Euler angles can be seen as a matrix chain multiplication of successive elemental rotations.

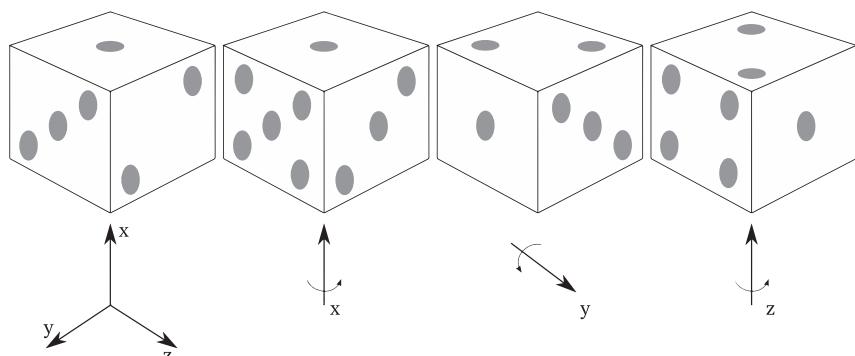


Figure 5.1 Euler angles intrinsic rotations on a six-faced dice.

Thus, they are taken in reverse order than executed: the first rotation is on the right and the last is on the left side, since successive rotations accumulate on the left side. Taking ϑ_x , ϑ_y , and ϑ_z , the Euler angles triplet for x - y - z rotation, and recalling the elemental matrix for simple rotations discussed in [Chapter 2](#)— Reference Systems and Planetary Models, we get:

$$\mathbf{A} = \begin{bmatrix} \cos \vartheta_z & \sin \vartheta_z & 0 \\ -\sin \vartheta_z & \cos \vartheta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \vartheta_y & 0 & -\sin \vartheta_y \\ 0 & 1 & 0 \\ \sin \vartheta_y & 0 & \cos \vartheta_y \end{bmatrix} \\ \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \vartheta_x & \sin \vartheta_x \\ 0 & -\sin \vartheta_x & \cos \vartheta_x \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \cos \vartheta_y \cos \vartheta_z & \cos \vartheta_x \sin \vartheta_z + \sin \vartheta_x \sin \vartheta_y \cos \vartheta_z & \sin \vartheta_x \sin \vartheta_z - \cos \vartheta_x \sin \vartheta_y \cos \vartheta_z \\ -\cos \vartheta_y \sin \vartheta_z & \cos \vartheta_x \cos \vartheta_z - \sin \vartheta_x \sin \vartheta_y \sin \vartheta_z & \sin \vartheta_x \cos \vartheta_z + \cos \vartheta_x \sin \vartheta_y \sin \vartheta_z \\ \sin \vartheta_y & -\sin \vartheta_x \cos \vartheta_y & \cos \vartheta_x \cos \vartheta_y \end{bmatrix}$$

It shall be noted that the rotation matrix around the second axis, y , has opposed sign with respect to the other two rotations since it is performed aligning the z axis on the x axis. This results in a shift in the signs of the matrix elements, as the reader might want to easily verify by deriving the three intermediate reference frame rotations.

Extrinsic rotations (i.e., rotations about the axes xyz of the original coordinate system, which is assumed to remain inertially fixed) follow the same rules and mathematical formulations of the intrinsic ones. The only difference is the way in which the elemental rotations are named and associated to an axis. Moreover, any extrinsic rotation is equivalent to an intrinsic rotation by the same angles but with inverted order of elemental rotations and vice versa (e.g., with reference to [Fig. 5.1](#), the equivalent extrinsic rotations would have been z - y - x).

The determination of the Euler angles from the DCM is mathematically cumbersome, as one would need to take care of inverse trigonometry and multiple solutions. Moreover, the inverse transformation is not always possible since a singularity condition exists, and it is usually denoted as *gimbal lock* condition [1]. The different sets of Euler angles are singular for different values of the second rotation angle, ϑ_2 , depending on the fact that the indexes are all different (i.e., Tait–Bryan angles), or the first and third indexes

coincide (i.e., proper Euler angles). In the Tait–Bryan angles, the singularity condition is $\vartheta_2 = (2n + 1)\pi/2$, while for the proper Euler angles, the singularity occurs when $\vartheta_2 = n\pi$. Therefore, Euler angles could be easily used in GNC functions if the spacecraft dynamics were always bounded with rotations far from the singularity conditions. However, since the spacecraft motion is generally not bounded, the attitude kinematics should be able to switch between a proper Euler angles sequence and a Tait–Bryan angles sequence in order to avoid the singularity conditions on ϑ_2 .

From these remarks, it is evident that Euler angles have a clear physical meaning, and they are a minimal attitude parametrization having only three independent components. They are very convenient for visualization purposes, but they are numerically time-consuming and have singularities not suitable for GNC applications. Moreover, the rule of consecutive rotations, although existing, is hard to be implemented due to its complexity.

Euler axis and angle

With reference to Fig. 5.1, we can further notice that the same result (i.e., 1 and 2 switch places and 3 goes to the opposite side) could have been obtained with a 180 degrees rotation about an axis perfectly aligned with the bisector plane between face 1 and 2 and parallel to face 3. This result is quite general since any number of consecutive rotations around any axis can be expressed in terms of a single rotation around a single axis. The axis and the angle take the name of *Euler axis* and *Euler angle*. The naming is not casual, as this is often referred to as Euler's theorem.

From matrix algebra, it is known that real, orthogonal matrices have one unit eigenvalue, to which we associate the eigenvector \mathbf{e} :

$$\mathbf{A}\mathbf{e} = 1\mathbf{e}$$

Therefore, vector \mathbf{e} does not change due to the rotation represented by matrix \mathbf{A} . This is possible only if the rotation occurs around axis \mathbf{e} , which is the Euler axis. The rotation amplitude around this axis is called Euler angle, ϕ .

The number of components to represent the spacecraft attitude using the Euler axis and angle is four, and thus this is not a minimal attitude representation. Indeed, one component is not independent given the unit norm of the Euler axis.

The DCM can thus be expressed in terms of the Euler axis $\mathbf{e} = \{e_1 \ e_2 \ e_3\}^T$ and Euler angle ϕ as follows:

$$\mathbf{A} = \mathbf{I}_3 \cos \phi - \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \sin \phi$$

$$+ (1 - \cos \phi) \begin{bmatrix} e_1^2 & e_1 e_2 & e_1 e_3 \\ e_1 e_2 & e_2^2 & e_2 e_3 \\ e_1 e_3 & e_2 e_3 & e_3^2 \end{bmatrix}.$$

The inverse operation is also possible, given the rotation matrix \mathbf{A} evaluate Euler axis and angle as:

$$\phi = \cos^{-1} \left[\frac{1}{2} (\text{tr}(\mathbf{A}) - 1) \right]$$

$$\begin{bmatrix} e_1 = \frac{A_{23} - A_{32}}{2 \sin(\phi)} \\ e_2 = \frac{A_{31} - A_{13}}{2 \sin(\phi)} \\ e_3 = \frac{A_{12} - A_{21}}{2 \sin(\phi)} \end{bmatrix}$$

where A_{ij} is the component of the \mathbf{A} matrix at the i -th row and j -th column.

It is remarked that the direct transformation, from Euler axis and angle to the attitude matrix, is always possible, while the inverse transformation is not always defined. The Euler axis is not defined when $\sin(\phi) = 0$, which corresponds to $n \times 180$ degrees rotations. In fact, there might be two 180 degrees rotations around different axes with the same effect, resulting in the singularity condition for the Euler axis and angle representation.

It is also remarked that the inverse trigonometric function \cos^{-1} returns two possible values of the Euler angle (i.e., $\pm \phi$), which consequently provide two different axes (i.e., $\pm \mathbf{e}$). Note that this is actually different from the singularity condition, since we are always looking at the same axis, once from the positive and once from the negative direction, and to each condition we associate the same rotation angle with opposite sign. This is physically the same condition.

Furthermore, the reader can verify that this form respects the periodicity of the angle, as any addition of 360 degrees would give the same result. This implies that given an axis, there are infinite angles possible, and thus when

inverting the operation, one must set some limits like (0 degrees, 360 degrees) or (-180 degrees, +180 degrees).

Any rotation composed by a generic number of elemental rotations is always referred to a Euler axis and angle. This is the building block of attitude kinematics, since if we “sum” several rotations, we end up with a final axis and angle representing the whole chain. However, this can be done only a-posteriori given the final and initial conditions, since there is no easy way to determine the accumulation process for the sequence of rotations.

Quaternions

Since it is not straightforward to accumulate successive rotations using only Euler axis and angle parametrization (i.e., without passing from the DCM), and because the existence of the singularity condition, a closely related object comes into play: unitary quaternions.

Quaternions have been introduced by William Rowan Hamilton in 1843 and are a mathematical tool that extend the complex algebra into three-dimensions, whose rule of multiplication happens to have a great connection with SO(3). Thus, they are powerful to represent rotations and to develop attitude kinematics for GNC applications. In fact, it has been found that unitary quaternions, \mathbf{q} , can be used to represent rotations and can be quite effective and efficient. It is possible to link a unitary quaternion with the Euler angle and Euler axis with the following formula:

$$\mathbf{q} = \begin{Bmatrix} \mathbf{q}_{\text{E3}} \\ q_4 \end{Bmatrix} = \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{Bmatrix} = \begin{Bmatrix} \mathbf{e} \sin \frac{\vartheta}{2} \\ \cos \frac{\vartheta}{2} \end{Bmatrix}$$

where we have used the convention that puts the scalar quaternion component, q_4 , in the fourth position, which is more comfortable when dealing with rotations. Mathematically, the scalar quaternion component is the real part of the quaternion and would be usually placed first, also according to the quaternion definition made by Hamilton with the scalar component at the beginning of the vector. Note that in this case, the scalar component is usually denoted as q_0 .

It is worth noticing here that, depending on the textbook or the application, the quaternion definition may vary. Unfortunately, the derived formulas and the associated mathematical operations are different according to

the used convention. Thus, we shall be very careful in understanding which quaternion convention shall be used in agreement with the formulas that are in use. A mistake in the selection of the proper quaternion convention to use (i.e., scalar first or scalar last) will induce computation errors leading to wrong GNC execution. The scalar last convention used in this book derives from the heritage of classical spacecraft attitude references, and it is very frequent in spacecraft GNC applications. This is not true for many different applications, and some computing libraries dealing with quaternion algebra use the quaternion first convention. The reader is strongly encouraged to always verify and spend time on the quaternion convention that is in use before using any third-party function or library.

The unitary condition that allows quaternions to represent attitude is:

$$\mathbf{q}^2 = \mathbf{q}_{13}^T \mathbf{q}_{13} + q_4^2 = 1 = q_1^2 + q_2^2 + q_3^2 + q_4^2 = \mathbf{e}^T \mathbf{e} \left(\sin \frac{\vartheta}{2} \right)^2 + \left(\cos \frac{\vartheta}{2} \right)^2$$

The number of components to represent the spacecraft attitude using quaternions is four, and thus also this parametrization is not a minimal attitude representation. Only three quaternion components are associated to the independent rotation parameters, and they can be used to perform the calculations. The fourth component can be always obtained imposing the unit norm quaternion constraint.

Through the quaternion definition, it is possible to obtain the relation with the DCM as follows:

$$\mathbf{A} = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1 q_2 + q_3 q_4) & 2(q_1 q_3 - q_2 q_4) \\ 2(q_1 q_2 - q_3 q_4) & 1 - 2(q_1^2 + q_3^2) & 2(q_2 q_3 + q_1 q_4) \\ 2(q_1 q_3 + q_2 q_4) & 2(q_2 q_3 - q_1 q_4) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix}$$

To recover the quaternions from the DCM, there are few possibilities like passing from the Euler axis or using directly the components of the DCM. In the latter case, one should be aware that it needs to consider cases where one or more components of the quaternion are null or close to zero. However, quaternions have no singular condition, since if a component vanishes, it is always possible to compute the inverse relations by exploiting different DCM components, which will be different from zero due to the normalization constraint. For example, if q_4 is different from zero, we may write:

$$\left[\begin{array}{l} q_1 = \frac{A_{23} - A_{32}}{4q_4} \\ q_2 = \frac{A_{31} - A_{13}}{4q_4} \\ q_3 = \frac{A_{12} - A_{21}}{4q_4} \\ q_4 = \pm \frac{1}{2}(1 + A_{11} + A_{22} + A_{33})^{\frac{1}{2}} \end{array} \right]$$

The inverse relations for the other components obtained from the main diagonal of \mathbf{A} can be found in many literature references [1,4]. In order to minimize the numerical errors in the GNC functions, we may select the one with the maximum value associated to the term with the sign ambiguity (i.e., \pm).

Note that the sign ambiguity has the same explanation given for the Euler axis and angle case. We are always looking at the same axis, once from the positive and once from the negative direction, and to each condition, we associate the same rotation angle with opposite sign. Thus, a change in sign of the quaternion is transparent for the physical attitude representation: $+\mathbf{q}$ and $-\mathbf{q}$ represent the same orientation in space. To avoid numerical problems and feedback control issues (e.g., *unwinding* phenomena), it is suggested to always enforce numerical continuity in the quaternion values. Namely, if the GNC functions would set to output an abrupt sign change in the quaternions, it is always better to manually change the sign of the new quaternion values to preserve the continuity with the previous ones.

Successive rotations

The most useful property of unitary quaternions applied to rotations is a multiplication rule that allows to obtain another quaternion representing the overall rotation resulting from two consecutive elemental rotations. The quaternion group is closed with respect to this multiplication meaning that multiplying two unitary quaternions leads to another unitary quaternion. As usual, the multiplicative formula to express a sequence of two consecutive rotations combines the quaternion components in the reversed order of rotations. Given a first rotation represented by quaternion \mathbf{q} , a second rotation represented by quaternion \mathbf{q}' , the overall rotation is represented by quaternion \mathbf{q}'' given by:

$$\begin{aligned}\mathbf{q}'' &= \mathbf{q}' \times \mathbf{q} = \left(\begin{bmatrix} -[\mathbf{q}'_{13}] & \mathbf{q}'_{13} \\ -\mathbf{q}'_{13}^T & 0 \end{bmatrix} + \mathbf{I}_4 q_4 \right) \mathbf{q} \\ &= \begin{bmatrix} q'_4 & q'_3 & -q'_2 & q'_1 \\ -q'_3 & q'_4 & q'_1 & q'_2 \\ q'_2 & -q'_1 & q'_4 & q'_3 \\ -q'_1 & -q'_2 & -q'_3 & q'_4 \end{bmatrix} \mathbf{q}\end{aligned}$$

where $[\mathbf{q}'_{1:3}]$ is the cross-product matrix containing the components of the vector part of the quaternion, \mathbf{I}_4 is the 4×4 identity matrix, and it has been used in the matrix form of the quaternion multiplication operation.

As one would expect, it is possible to chain any number of rotations in the same way we chained DCM for successive rotations. The operation can be easily inverted using the inverse operation that takes the form of a slightly different matrix [1]. It should be noted that we can always express the inverse rotation using the conjugate of the quaternion, where only the vector part, $\mathbf{q}_{1:3}$, is changed in sign: this is the easiest way to invert a quaternion rotation. Another interesting aspect that can be exploited is that there exists another matrix form where the two original quaternions, \mathbf{q} and \mathbf{q}' , can be swapped without changing the results [1].

Relative quaternion

Quaternion multiplication allows to define the relative quaternion between two different attitude states represented by two quaternions. This operation is very useful to define relative attitude states and to define the error quaternion with respect to a reference state. The latter is very useful in attitude estimation and control to define an error difference relative to a desired attitude state.

The relative quaternion can be defined as:

$$\delta \mathbf{q} = \begin{Bmatrix} \delta \mathbf{q}_{13} \\ \delta q_4 \end{Bmatrix} = \mathbf{q} \times \mathbf{q}_{ref}^{-1} = \begin{Bmatrix} \chi(\mathbf{q}_{ref}) \mathbf{q} \\ \mathbf{q}_{ref}^T \mathbf{q} \end{Bmatrix},$$

where $\chi(\mathbf{q}_{ref})$ is a 3×4 matrix defined as:

$$\chi(\mathbf{q}_{ref}) = [\mathbf{I}_3 \ q_{ref4} \ -[\mathbf{q}_{ref1:3}] \ -\mathbf{q}_{ref1:3}].$$

If $\delta \mathbf{q}$ is normalized and it is close to the identity quaternion, then $\delta \mathbf{q}_{1:3} \approx \boldsymbol{\alpha}/2$ and $\delta q_4 \approx 1$, where $\boldsymbol{\alpha}$ is a vector of small angle rotations. As \mathbf{q} approaches \mathbf{q}_{ref} , then $\delta \mathbf{q}_{1:3}$ and $\boldsymbol{\alpha}$ both approach zero.

It shall be remarked that the quaternions are not the unique mathematical parametrization that can connect with Euler axis and angle. Some alternatives like the Gibbs and Rodrigues vectors exist [1,4]. However, the quaternions have a larger number of advantages with respect to the other attitude parametrizations and to the drawbacks. For this reason, they are widely used in GNC applications, and they are the go-to formulation in most space engineering references.

Few important aspects of quaternions made them appealing for the GNC community: absence of trigonometric functions for most applications, absence of singularity in critical areas, possibility to easily chain successive rotations, easy normalization, and continuity numerical verification. Essentially, quaternions are more computationally friendly than other attitude parametrizations. Their main drawback is related to their poor physical meaning, and, therefore, their use is not intuitive. However, this is a problem for the human user and not for the GNC system. Typically, a simple conversion from quaternions to Euler angles solves this problem and allows to visualize and understand the spacecraft attitude.

Further details about quaternion algebra can be found in the appendix [Chapter 16](#)—Mathematical and Geometrical Rules. Moreover, the reader is invited to deepen study about quaternion properties and rules in Ref. [1].

Attitude variation in time

Previous sections discussed the way in which the orientation of a spacecraft in space is represented and mathematically parametrized. Now it is time to address how attitude changes in time, introducing the concept of the *angular velocity* or *angular rate* vector.

Intuitively, we can think about the angular velocity by considering a disk rotating around its normal axis. We know that the variation in time of the angle that describes the system is equal to the velocity of rotation. In general, the angular velocity is represented by a vector that is characterized by a direction and an intensity. In the disk case, we have that the angular velocity can assume one orientation in space and infinite positive or negative intensities. Conventionally, the positive direction is taken counterclockwise, and if one uses the right-hand rule, we can easily see that taking the vector orientation as the thumb leaves the fingers pointing in the rotation direction. If

another disk is mounted on the first one with a different axis of rotation, then the velocity of any points of its surface requires the knowledge of the velocity of the first disk. If we chain three disks with perpendicular axes, we obtain what is often referred to as a mechanical gyroscope, figured in Fig. 5.2, and the overall velocity vector shall be described in three dimensions, given the fact that now infinite orientations in space are possible.

More formally, the attitude variation in time can be described considering the time variation of the fundamental attitude representation:

$$\dot{\mathbf{A}}(t) = \lim_{\Delta t \rightarrow 0} \frac{\mathbf{A}(t + \Delta t) - \mathbf{A}(t)}{\Delta t} = \lim_{\Delta t \rightarrow 0} \frac{-[\Delta_b \boldsymbol{\theta}_{IJK}^b] \times}{\Delta t} \mathbf{A}(t),$$

where we remind that $\mathbf{A} = {}_b\mathbf{A}_{IJK}$ describes the rotation from the inertial frame to the spacecraft body one, and $\Delta_b \boldsymbol{\theta}_{IJK}^b$ is a small angle rotation vector that is contained inside the cross-product matrix. The rotation is performed in the interval of time Δt . Note that the subscripts b and IJK mean that the small angle rotation is related to the rotation from frame IJK to frame b , and the superscript b means that the rotation angle is represented in body frame b .

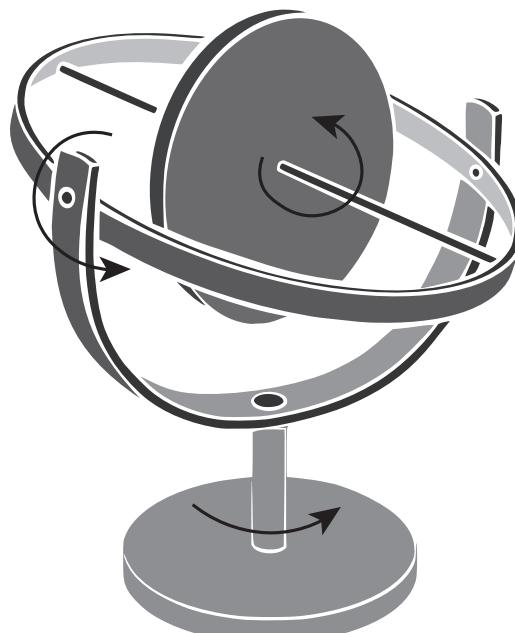


Figure 5.2 Gyroscope.

The rotation angle is small since the time variation is computed in the limit that Δt goes to 0; it is expressed in b because the difference $\mathbf{A}(t + \Delta t) - \mathbf{A}(t)$ can be represented by a differential rotation from frame b at time t to frame b at time $t + \Delta t$, and these two frames coincide for Δt going to 0. Moreover, the previous kinematic relation does not distinguish between the situations where frame IJK or frame b or both frames are rotating in an absolute sense; it only cares about the relative rotation between the two frames.

Angular velocity

At this point, we can define the angular velocity vector:

$${}_b\boldsymbol{\omega}_{IJK}^b(t) \equiv \lim_{\Delta t \rightarrow 0} \frac{\Delta_b \boldsymbol{\theta}_{IJK}^b}{\Delta t},$$

noting that the angular rate is related to the rotation from a reference frame to another one, and it is always expressed in one reference frame. In this case, the angular velocity describes the time variation of the rotation from frame IJK to frame b , and it is mathematically expressed with the vector components in the frame b . It shall be noted that the physical quantity does not depend on the frame in which it is expressed. Indeed, the angular velocity vector represents a physical vector in space relating instantaneous successive rotations of a reference frame with respect to another, but it can be expressed in whichever reference frame. For example, we can mathematically express the previous angular velocity vector in the inertial frame as:

$${}_b\boldsymbol{\omega}_{IJK}^{IJK} = {}_{IJK}\mathbf{A}_b {}_b\boldsymbol{\omega}_{IJK}^b.$$

The relation between the two vectors is given by the DCM whose variation in time is a function of the angular velocity itself. The fact that the magnitude of the angular velocity does not change when looking at it from the two frames is intrinsic in the orthonormality property of the DCM. De facto, the norm of the angular velocity between two frames is a scalar and thus invariant between two frames.

In order to simplify the discussion, we will omit the subscripts, unless it is necessary to specify the reference frames involved in the attitude kinematics. In general, we will consider the rotation from frame IJK to frame b as the nominal direct rotation (i.e., the rotation of the spacecraft body frame with respect to the inertial one: $\mathbf{A} = {}_b\mathbf{A}_{IJK}$). Therefore, we can rewrite the previous expression as:

$$\boldsymbol{\omega}^{IJK} = \mathbf{A}^T \boldsymbol{\omega}^b.$$

The fundamental kinematics relation can be written in terms of the angular velocity as:

$$\dot{\mathbf{A}}(t) = \lim_{\Delta t \rightarrow 0} \frac{-[\Delta_b \boldsymbol{\theta}_{IJK}^b]}{\Delta t} \mathbf{A}(t) = -[\boldsymbol{\omega}^b(t)_\times] \mathbf{A}(t).$$

With some algebraic manipulations, we can also use the angular velocity expressed in inertial frame as:

$$\dot{\mathbf{A}}(t) = -[\boldsymbol{\omega}^b(t)_\times] \mathbf{A}(t) = -\mathbf{A}(t) [\boldsymbol{\omega}^{IJK}(t)_\times].$$

Moreover, we can derive the kinematics for the inverse rotation from the frame b to frame IJK , by taking the transpose of the fundamental kinematics relation and remembering that the cross-product matrix is skew symmetric:

$$\dot{\mathbf{A}}^T(t) = \mathbf{A}^T(t) [\boldsymbol{\omega}^b(t)_\times] \Leftrightarrow \dot{\mathbf{A}}_{bIJK} = \mathbf{A}_{bIJK}(t) [\boldsymbol{\omega}_{IJK}^b b(t)_\times]$$

The frame labels are always arbitrary, so we can exchange the labels IJK and b in this expression to get:

$$\dot{\mathbf{A}}_{IJK_b} = \mathbf{A}_{IJK_b}(t) [\boldsymbol{\omega}_b^{IJK} IJK(t)_\times] \Leftrightarrow \dot{\mathbf{A}}(t) = \mathbf{A}(t) [\boldsymbol{\omega}_b^{IJK} IJK(t)_\times].$$

Hence, comparing this expression with the fundamental kinematics equation in inertial frame, we can see that:

$$_b \boldsymbol{\omega}_{IJK}^{IJK} = - {}_{IJK} \boldsymbol{\omega}_b^{IJK},$$

where we found the obvious result that the rotational rate of frame IJK with respect to frame b is the negative of the rate of frame b with respect to frame IJK . Although we derived this as a relationship of the components in frame IJK , the properties of orthogonal transformations show that it must be true for the components in any frame:

$$_b \boldsymbol{\omega}_{IJK}^b = - {}_{IJK} \boldsymbol{\omega}_b^b.$$

Angular velocities can be summed, and they have all the common vector properties. However, we must take care of the reference frame in which the addition operation is performed. To make an example, let's take three reference frames, p , q , and r , which are in relative motion with angular velocities $_r \boldsymbol{\omega}_q$ and $_q \boldsymbol{\omega}_p$. We can compute the overall angular velocity between r and p , expressed in the final reference frame, r , as:

$$_r \boldsymbol{\omega}_p^r = {}_r \boldsymbol{\omega}_q^r + {}_r \mathbf{A}_q \left({}_q \boldsymbol{\omega}_p^q \right) = {}_r \boldsymbol{\omega}_q^r + {}_q \boldsymbol{\omega}_p^r,$$

The central side of this equation is more useful in applications, since we usually have the angular rates measured in different reference frames. But the right-hand side shows that the angular velocity of the motion of frame r relative to frame p is just the vector sum of angular velocity of the motion of r relative to q , and the angular velocity of the motion of q relative to p , provided that they are expressed in the same reference frame. In this case, they are all expressed in r , but the result is valid in any existing reference frame.

Euler angles kinematics

Specializing the attitude kinematics for different attitude parametrizations, we can see how the angular velocity relates to Euler angles and quaternions. Looking again at the mechanical gyroscope in Fig. 5.2, it is possible to see that the way the disks are mounted closely resembles the three rotations of the Euler angles triplets. The variation in time of the Euler angles can be related to the angular velocity of the body by differentiation of the Euler angles relations, obtaining the following expression for the x - y - z sequence:

$$\boldsymbol{\omega}^b = \begin{Bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{Bmatrix} = \begin{bmatrix} \cos \vartheta_y \cos \vartheta_z & \sin \vartheta_z & 0 \\ -\cos \vartheta_y \sin \vartheta_z & \cos \vartheta_z & 0 \\ \sin \vartheta_y & 0 & 1 \end{bmatrix} \begin{Bmatrix} \dot{\vartheta}_x \\ \dot{\vartheta}_y \\ \dot{\vartheta}_z \end{Bmatrix}$$

Inverting the relation gives:

$$\begin{Bmatrix} \dot{\vartheta}_x \\ \dot{\vartheta}_y \\ \dot{\vartheta}_z \end{Bmatrix} = \frac{1}{\cos \vartheta_y} \begin{bmatrix} \cos \vartheta_z & -\sin \vartheta_z & 0 \\ \cos \vartheta_y \sin \vartheta_z & \cos \vartheta_y \cos \vartheta_z & 0 \\ -\sin \vartheta_y \cos \vartheta_z & \sin \vartheta_y \sin \vartheta_z & \cos \vartheta_y \end{bmatrix} \boldsymbol{\omega}^b$$

Such relation is nonlinear and suffers from a singularity. When $\vartheta_y = \pm 90$ degrees, we have that the derivative is infinite and any hope to relate the two velocities is lost. This condition is related with the gimbal lock phenomenon, which is also present in mechanical gyroscopes: when this occurs, the two Euler angles ϑ_x and ϑ_z insist on the same axis and the DCM becomes a rotation matrix around axis y function of an angle $\vartheta_x + \vartheta_z$.

For all these reasons, Euler angles are not convenient for spacecraft GNC application, and they should be used only for visualization purposes. In fact, if we want to use Euler angles for computation, we must bear in mind that

every Euler angle sequence has a singularity. Euler angles are popular in aircraft GNC applications, since the singularity problem might be overcome by the fact that the envelope of possible rotations for an airplane is usually limited; this is not true for satellites that usually span the whole rotation space.

Quaternions kinematics

Quaternion offers a simpler and singularity-free relation between quaternion derivative and angular velocity of the reference frame:

$$\dot{\mathbf{q}} = \frac{1}{2} \begin{bmatrix} -[\boldsymbol{\omega}_x^b] & \boldsymbol{\omega}^b \\ -\boldsymbol{\omega}^{bT} & 0 \end{bmatrix} \mathbf{q}$$

The only issue in using this formulation to integrate the attitude evolution in time is that the operation of numerical integration does not enforce the unitarity of the quaternion. Then, at each numerical integration step, the quaternion normalization shall be enforced. This is mathematically trivial and simpler than in the case of DCM. In fact, the direct integration of the DCM derivative does not preserve the DCM properties, and orthonormality must be enforced at each integration step. For this and all the other reasons discussed in this section, quaternions are commonly preferred for computation in any GNC system.

It is often more convenient to integrate the quaternion kinematics in a different form:

$$\dot{\mathbf{q}} = \frac{1}{2} \begin{bmatrix} \mathbf{I}_3 q_4 + [\mathbf{q}_{t3} \times] \\ -\mathbf{q}_{t3}^T \end{bmatrix} \boldsymbol{\omega}^b,$$

which can also be easily inverted to obtain the angular velocity as a function of the derivative of the quaternion:

$$\boldsymbol{\omega}^b = 2 \|\mathbf{q}\|^{-2} [\mathbf{I}_3 q_4 - [\mathbf{q}_{t3} \times] - \mathbf{q}_{t3}] \dot{\mathbf{q}},$$

where the norm of the quaternion is unitary, $\|\mathbf{q}\| = 1$. Thus, it is overabundant in the previous equations and can be omitted.

It should be noted that the previous equations link the derivative of the quaternion with respect to the angular velocity described in one of the two frames (i.e., the body frame). It is entirely possible to rewrite the formula in terms of the angular velocity expressed in the other reference frame [1]. Then, it is an arbitrary choice the linking of the quaternion to that angular velocity in that way and, in general, the convention used to

link a quaternion and the angular velocity must be consistent and specified, as few variants are possible. More details are available in the appendix [Chapter 16](#) – Mathematical and Geometrical Rules, and in the suggested references [1].

In this book, the notation \mathbf{q} is associated to the DCM, \mathbf{A} . Hence, this quaternion represents the rotation of the spacecraft body frame with respect to the inertial one (i.e., the rotation from IJK to b). However, the reader is invited to note that this is the not unique possibility, and other assumptions must be always declared.



Attitude dynamics

We now focus the discussion on attitude dynamics, first defining the inertia parameters influencing the attitude equations of motion and then deriving them from the fundamental mechanical quantity in rotational dynamics: the angular momentum. The section also contains some discussion on attitude stability and attitude dynamics in spinning conditions. Finally, the contributions of environmental torques in the attitude dynamics is mathematically discussed.

Inertia matrix

When we study the motion of a rigid body, we typically define one fundamental physical property that is called mass and it is usually sufficient to describe the inertia of the body along its linear motion. When we talk about rigid body rotations, this is no longer enough, as a rigid body is never a point mass, but a continuous distribution of mass. Different mass distributions give different inertia properties to objects, and they have impact in the way they are moving in space.

Let us assume a scalar function $\rho(\mathbf{r})$ that denotes the density of the rigid body, and it is solely a function of the spatial coordinate \mathbf{r} of a reference frame attached to the body. For now, let us assume that this function is continuous and can characterize the whole body. Later, we will extend the concept in a simpler engineering way.

Given the density function, we can determine the system mass using an integral over the volume V of the object.

$$m = \int_V \rho(\mathbf{r}) dV$$

Now we can determine the first (**c**) and second (**I**) moment of inertia as follows:

$$\mathbf{c} = \int_V \mathbf{r} \rho(\mathbf{r}) dV$$

$$\mathbf{I} = \int_V (\mathbf{r}^2 \mathbf{I}_3 - \mathbf{r} \otimes \mathbf{r}) \rho(\mathbf{r}) dV$$

where the symbol \otimes represents the tensor product, or outer product, between two vectors. The second moment of inertia is the most important for attitude dynamics, and it is usually referred to as *inertia tensor* or *inertia matrix*.

The inertia tensor is a 3×3 symmetric matrix. The terms along the diagonal are called *inertia moments* or *moments of inertia*, while the off-diagonal terms are called *inertia products*. Inertia moments are always positive, while there is no general rule for the inertia products. The reader should consider that the term *moment of inertia* can be used in other engineering fields with slightly different meanings, often referring to the component of the inertia tensor about the rotation axis in use, or to the largest inertia tensor component.

The inertia tensor has been constructed from the origin of a reference frame along its axes. This inherently means that differently oriented frames or with different origins would give different results; thus, we must specify the origin and the reference axes used to compute the inertia matrix. To convert the inertia tensor from a reference frame to a different one, we could substitute the rototranslation transformations, needed to relate two generic reference frames, and obtain a complex formula. However, there are some engineering tricks that simplify the problem and can be rather easy to be applied. For these engineering solutions, we assume that the density of the rigid body is constant $\rho(\mathbf{r}) = \rho$.

The first trick is the change in orientation of the reference frame to compute the inertia matrix. If we pre- and post-multiply the inertia tensor by a DCM and its transpose, we can express the body moment of inertia in any reference frame with the same origin:

$$\mathbf{I}_2 = {}_2\mathbf{A}_1 \mathbf{I}_1 {}_2\mathbf{A}_1^T = {}_2\mathbf{A}_1 \mathbf{I}_1 {}_1\mathbf{A}_2$$

The second is the translation of the origin making use of the parallel axis theorem, which is sometimes referred to with the names of Huygens and

Steiner. With this theorem, we can translate the moment of inertia from one reference to a parallel one by using the origin's displacement vector \mathbf{r}_{12} :

$$\mathbf{I}_2 = \mathbf{I}_1 + m(\|\mathbf{r}_{12}\|^2 \mathbf{I}_3 - \mathbf{r}_{12} \otimes \mathbf{r}_{12}) = \mathbf{I}_1 + m(\|\mathbf{r}_{12}\|^2 \mathbf{I}_3 - \mathbf{r}_{12} \mathbf{r}_{12}^T).$$

The reader might be more familiar with the moment of inertia transport around a single axis. Assuming a translation orthogonal to z with displacement $\mathbf{d} = \{d_x \ d_y\}^T$, the parallel axis theorem reads:

$$I_{z2} = I_{z1} + m d^2 = I_{z1} + m(d_x^2 + d_y^2)$$

Now we have all the tools needed to address the inertia matrix in any reference frame; however, depending on the situation, there are some reference frames that are more useful than others.

Looking at the definition of the first moment of inertia, we can clearly see that there is a specific origin of the reference frame where the first moment of inertia is null. This represents the rotational center of the body, and if we leave the body spinning in an empty space, this point will be the only point not changing position in time. This is the *Center of Mass* (*CoM*) of the body. Under a different perspective, we can also state that such a point minimizes the inertia of the system and would take less effort to spin the body if it rotates around that point.

Defining the body reference frame's origin in the CoM of the spacecraft is common practice when dealing with satellites, as we will see later; but this is not the only choice we can make for the body reference frame. Recalling the property of rotation of the inertia tensor, we have seen that we can rotate the reference frames in any direction; this means that we can also find a particular orientation for which the inertia matrix is diagonal. This is true because the inertia matrix is symmetric and positive definite, so we can always find an orientation that exposes its eigenvalues along the main diagonal. Thus, we can solve a standard eigenvalue problem for the matrix \mathbf{I} , finding a rotation matrix whose rows are composed by the eigenvectors associated to the eigenvalues of \mathbf{I} . The found reference frame is aligned with the *principal axes of inertia* of the body, and it will be used later to simplify dynamical equations. The principal inertia frame is therefore a reference frame with origin in the center of mass of the rigid body and such that all inertia products are equal to zero. The elements on the main diagonal are the eigenvalues of the inertia tensor, and they are referred to as principal moments of inertia, I_x , I_y , and I_z .

Rigid body dynamics

Spacecraft attitude dynamics is based on the theory of rigid body dynamics, since the spacecraft can be typically modeled as a rigid body. If some flexible elements are present, or the spacecraft is inherently flexible, the attitude dynamics can include the flexible terms as discussed in the [Chapter 3 – The Space Environment](#).

Angular momentum

Angular momentum is a vector quantity that connects the angular velocity of a rigid body and its moments of inertia, pretty much like linear momentum links mass and velocity of a rigid body. Angular momentum is a conserved quantity, since the total angular momentum of a closed system (i.e., no external torque is applied) remains constant. Torque can be defined as the rate of change of angular momentum, analogous to force for the linear momentum. In these regards, the inertia matrix has a key role in this, as two bodies with different moments of inertia would need different torques to achieve the same variation of the angular rate. This can intuitively be represented by a light wooden door and a security fire barrier door: the lighter one requires less effort to be opened or closed, since it has a lower moment of inertia with respect to the hinges. Namely, the larger the inertia values, the higher the required torque to impose a rotation to an object.

The angular momentum of a rigid body b is defined referring to a reference frame attached to the rigid body and moving with it. Indeed, unlike linear momentum, angular momentum depends on where the origin is chosen. Assuming a body-fixed reference frame with origin, O , we can compute the angular momentum over the volume of the body as:

$$\mathbf{h}^b = \int_V \mathbf{r} \times \rho(\mathbf{r}) \mathbf{v} dV,$$

where \mathbf{v} is the velocity of any infinitesimal particle of the body. Solving the integral and using the definition of first and second moment of inertia of the body, the angular momentum can be expressed as:

$$\mathbf{h}^b = -\mathbf{v}_O \times \mathbf{c} + \mathbf{I} \boldsymbol{\omega}^b$$

where \mathbf{v}_O is the velocity of the origin of the reference system attached to the rigid body, and $\boldsymbol{\omega}^b$ the angular rate of the body frame with respect to the inertial one expressed in body frame.

By choosing the origin O coincident with the center of mass of the system, we know that \mathbf{c} vanishes and the angular momentum is simplified as:

$$\mathbf{h}^b = \mathbf{I} \boldsymbol{\omega}^b.$$

Rotational kinetic energy

Whenever there is an object in motion, we can associate it with the concept of energy, and in particular kinetic energy. As for the angular momentum, there exists the rotational counterpart of the linear kinetic energy. Analogously, the rotational kinetic energy is constant for a closed system with no internal energy dissipation. In fact, in the absence of external disturbances and torque fields capable of inducing energy to the system, kinetic energy is a scalar constant. This applies to extended bodies that can rotate in space.

The rotational kinetic energy, T_ω , is given by:

$$T_\omega = \frac{1}{2} \boldsymbol{\omega}^{b^T} \mathbf{I} \boldsymbol{\omega}^b,$$

where both the angular rate and the inertia matrix are computed in a reference frame with the origin in the center of mass of the system. In case of external torques, we could link them to the variation in time of the kinetic energy.

For a given inertia matrix, kinetic energy, and angular momentum of the body, we can define the so-called inertia ellipsoid, kinetic energy ellipsoid, and angular momentum ellipsoid [1,4]. A complete discussion on these geometrical entities is out of the scope of this book, and it can be found in literature. However, they are powerful visualization tools to understand and bound the attitude dynamics of a rigid body in space. In fact, the angular momentum ellipsoid must intersect the kinetic energy ellipsoid at least in one point in order to represent a real motion. If no energy dissipation exists and no torque is applied, both kinetic energy and angular momentum must be constant. For a given T_ω and \mathbf{h} , the attitude dynamics must evolve in such a way that angular velocity vector identifies the intersection of the kinetic energy ellipsoid with the angular momentum ellipsoid. Indeed, the angular velocity must be compatible with the energy level and with the angular momentum defined by the initial conditions of motion. The intersection of the kinetic energy ellipsoid with the angular momentum ellipsoid generates two lines, called *polhodes* [4]. This can be also visualized on the invariant plane, which is the fixed plane containing the terminal point of vector $\boldsymbol{\omega}^b$, because of the conservation of the projection of $\boldsymbol{\omega}^b$ over \mathbf{h} for a torque-free motion

with no energy dissipation. The line traced by the angular velocity on the invariant plane is in general not closed, it is called *herpolhode* [1].

From polhodes and herpolhode analyses, we derive the fundamental result that a stable rotational motion is possible only around the maximum or minimum principal moment of inertia axis. In fact, if the angular velocity is slightly shifted from the maximum moment of inertia axis or from the minimum moment of inertia axis, it will remain confined in a region close to the axis, while a slight shift from the intermediate moment of inertia axis would cause a dramatic departure from the original condition. It is a well-known result that spin stabilization should always be about the principal axis with the largest or smallest principal moment of inertia, known as the major or minor principal axis [1].

Note that the conservation of angular momentum occurs if no external forces are applied, but the kinetic energy is constant only if also no internal dissipation is present. In the case internal forces (e.g., due to flexibility) dissipate the system's energy, the rotational energy will decrease to its minimum value of $\mathbf{h}^2/I_{\text{Max}}$, resulting in a stable rotation about a major principal axis with moment of inertia I_{Max} . Thus, for practical applications, only the principal axis with maximum moment of inertia shall be used as a stable spinning axis of the spacecraft. If the desired spin axis is a minor axis, energy dissipation will induce an undesirable rotation about an axis perpendicular to the preferred one. This unwanted spin condition was unfortunately verified in history with the Explorer 1, the first Earth satellite launched by the United States.

Euler equation

We can exploit Lagrange formulation to derive a set of complete equations of motion exclusively for a rotating body in space. First, we set the fundamental equation for rigid body motion in inertial frame, i :

$$\frac{d\mathbf{h}^i}{dt} = \mathbf{t}^i,$$

where \mathbf{t}^i are the external torques expressed in inertial frame. However, it is more convenient to derive the dynamical equations as observed in the body reference frame. By using the attitude matrix, $\mathbf{A} = {}_b\mathbf{A}_i$, and deriving with respect to time, we get:

$$\mathbf{A}^T \frac{d}{dt} (\mathbf{I} \boldsymbol{\omega}^b) + \frac{d}{dt} \mathbf{A}^T \cdot (\mathbf{I} \boldsymbol{\omega}^b) = \mathbf{t}^i$$

Recalling the time variation equation for the DCM, $\dot{\mathbf{A}}^T = \mathbf{A}^T [\boldsymbol{\omega}_\times^b]$, we get:

$$\mathbf{A}^T \frac{d}{dt} (\mathbf{I} \boldsymbol{\omega}^b) + \mathbf{A}^T [\boldsymbol{\omega}_\times^b] \cdot (\mathbf{I} \boldsymbol{\omega}^b) = \mathbf{A}^T \mathbf{t}^b,$$

where we also used the external torques in the body frame, which are handier for GNC applications.

Finally, by simplifying the equation above, we get:

$$\frac{d}{dt} \mathbf{h}^b = \mathbf{t}^b - [\boldsymbol{\omega}_\times^b] (\mathbf{I} \boldsymbol{\omega}^b),$$

or in a simpler notation:

$$\dot{\mathbf{h}}^b + \boldsymbol{\omega}^b \times \mathbf{h}^b = \mathbf{t}^b$$

This formulation is often referred to as Euler equation and relates the angular momentum to the external torques applied to the body. It should be noted that there might be cases where the inertia of the system is not fixed in time and the time derivative of the angular velocity depends also on the time variation of the inertia of the system. In the case of constant inertia, the angular velocity of the spacecraft is directly related to the external torques as:

$$\dot{\boldsymbol{\omega}}^b = \mathbf{I}^{-1} [\mathbf{t}^b - \boldsymbol{\omega}^b \times \mathbf{I} \boldsymbol{\omega}^b].$$

Another important consideration of Euler equation is that in absence of external actions, a rigid body does not see its angular velocity magnitude change, but it might see its direction changing in time, thanks to the nonlinear term $\boldsymbol{\omega}^b \times \mathbf{h}^b$. In fact, without external torques, the kinetic energy of a rigid body with no internal dissipation is a scalar invariant of the system. We can look at it in the following way:

$$\frac{d}{dt} T_\omega = \boldsymbol{\omega}^{b^T} \frac{d}{dt} (\mathbf{h}^b) = \boldsymbol{\omega}_b^T [\mathbf{t}^b - [\boldsymbol{\omega}_\times^b] (\mathbf{I} \boldsymbol{\omega}^b)] = \boldsymbol{\omega}_b^T \mathbf{t}^b$$

since the dot product of a cross product is null. However, kinetic energy is scalar, and this means that the angular velocity direction can freely change in time.

As previously hinted, it is possible to simplify Euler equations if we express it in the principal axis reference frame. Let us use the common notation $\boldsymbol{\omega}^b = \{ \omega_x \ \omega_y \ \omega_z \}^T$, $\mathbf{t}^b = \{ t_x \ t_y \ t_z \}^T$ and $\mathbf{I} = \text{diag}(\{ I_x \ I_y \ I_z \})$, in order to get:

$$\begin{cases} I_x \dot{\omega}_x + (I_z - I_y)\omega_y\omega_z = t_x \\ I_y \dot{\omega}_y + (I_x - I_z)\omega_x\omega_z = t_y \\ I_z \dot{\omega}_z + (I_y - I_x)\omega_x\omega_y = t_z \end{cases}$$

In these equations, if the coupling term vanishes, then the rotation around one axis will be driven only by the torque around the same axis.

For the sake of simplicity and to gain some insight on the mathematical model, we can assume a torque-free motion, $\mathbf{t}^b = 0$, and rearrange the Euler equations in the principal inertia axes:

$$\begin{Bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{Bmatrix} = \begin{Bmatrix} \frac{I_y - I_z}{I_x} \omega_y \omega_z \\ \frac{I_z - I_x}{I_y} \omega_x \omega_z \\ \frac{I_x - I_y}{I_z} \omega_x \omega_y \end{Bmatrix} = \begin{Bmatrix} K_1 \omega_y \omega_z \\ K_2 \omega_x \omega_z \\ K_3 \omega_x \omega_y \end{Bmatrix}$$

where K_1 , K_2 , and K_3 are the so-called inertia ratios. This nonlinear formulation explains better the relation between the angular velocities of different axes. As an example, if we assume $I_x = I_y$, we have that $\dot{\omega}_z = 0$ and this transforms the first two line of the equation in two harmonic oscillators whose period is given by inertia ratio and the nominal angular velocity of the third axis. This is one of the few cases where an analytical solution exists [4]. Another would be the “sphere” case where the nonlinear term nullifies, thanks to the symmetry, resulting in a trivial dynamical solution. Even the case in which only one angular rate component is different from zero, results in a very simple dynamics and all the coupling terms are null. This case is called *simple spin*, and it is very relevant for practical applications and spin stabilization techniques [1]. However, the reader should notice that the simple spin stays with all the nonlinear terms equal to zero only in the ideal case; a small perturbation bringing the angular velocity also on the other principal axes would lead to a more complex triaxial dynamics.

Attitude dynamics of a satellite is then described by joining Euler equations for attitude dynamics, and an attitude parameterization to propagate the attitude kinematics. In fact, on a satellite, we have torques influencing directly the dynamics and induce changes in the components of angular velocity. By adopting one attitude parameterization, we can, knowing the angular velocity, calculate the attitude parameters that indicate the orientation of the satellite in space and that allow to evaluate the position-

dependent torques. In Euler equations, it is therefore required to assign initial conditions for angular velocity, while in attitude parameterization, we must assign the initial satellite attitude.

It is relevant to remark that the use of the principal axis reference frame is not particularly convenient for numerical integration of the attitude dynamics and for GNC applications, since a computer can easily deal with a full 3×3 inertia tensor. However, the principal reference frame is very useful for theoretical investigations on the attitude motion and for stability analyses.

Attitude stability

One important aspect when looking at the dynamics of a rotating body in space is the stability of rotation around a predefined angular velocity. To investigate stability, let us differentiate the torque-free motion equations in principal axes to obtain the equations of three coupled nonlinear harmonic oscillators:

$$\begin{cases} \ddot{\omega}_x = (K_1 K_2 \omega_z^2 + K_1 K_3 \omega_y^2) \omega_x \\ \ddot{\omega}_y = (K_2 K_3 \omega_x^2 + K_1 K_2 \omega_z^2) \omega_y \\ \ddot{\omega}_z = (K_1 K_3 \omega_y^2 + K_2 K_3 \omega_x^2) \omega_z \end{cases}$$

If we assume to have a nonzero reference velocity $\bar{\boldsymbol{\omega}}^b$ such that $\boldsymbol{\omega}^b = \bar{\boldsymbol{\omega}}^b + \delta\boldsymbol{\omega}^b$ and $\dot{\boldsymbol{\omega}}^b = \delta\dot{\boldsymbol{\omega}}^b$, we can study the dynamical stability of the system. To be stable, we must have $\delta\boldsymbol{\omega}^b$ always bounded; hence, the associated nonlinear oscillators must have the terms in brackets always negative. This highly depends on the inertia ratios (i.e., the signs of K_1 , K_2 , and K_3) and $\boldsymbol{\omega}^b$ itself. Hence, results may differ from one satellite to another due to inertia properties and mission profile. However, if we limit ourselves to the first order terms and assume $\bar{\boldsymbol{\omega}}^b$ aligned with one of the principal axes, we can find some interesting outcomes.

From [Table 5.1](#) representing three relevant perturbed simple spin cases, we can extract one important information: there will be always at least one unstable direction of rotation given any satellite. If we assume, for example, $I_x > I_y > I_z$, then we would have $K_1, K_3 > 0$ and $K_2 < 0$ meaning that rotation around the intermediate axis, I_y , would be unstable since $K_1 K_3 > 0$. This result is independent from the order of the principal inertias components, as the reader might want to check independently.

The stability analysis can be further expanded by taking into account also the kinematics of the satellite with quaternions. Let us restrict the reasoning to small variations around a reference configuration under a constant rotation (i.e., $\mathbf{q} = \{0, 0, 0, 1\}^T$). Let's apply a small perturbation such that $\mathbf{q}_{1:3}^T \mathbf{q}_{1:3} \ll q_4 \approx 1$, which represents small angular variation to the attitude. Limiting to the first order terms, we obtain the following expression, where the constant factor $1/2$ has been neglected because of the variational focus of this discussion:

$$\dot{\mathbf{q}}_{13} = \boldsymbol{\omega}^b q_4 - \boldsymbol{\omega}^b \times \mathbf{q}_{13} \approx \delta \boldsymbol{\omega}^b - \overline{\boldsymbol{\omega}}^b \times \mathbf{q}_{13}.$$

Then, differentiating with respect to time gives:

$$\ddot{\mathbf{q}}_{13} = \dot{\delta \boldsymbol{\omega}^b} - \dot{\overline{\boldsymbol{\omega}}}^b \times \dot{\mathbf{q}}_{13}.$$

If we expand to first order, recalling $\boldsymbol{\omega}^b = \overline{\boldsymbol{\omega}}^b + \delta \boldsymbol{\omega}^b$, and we substitute $\delta \boldsymbol{\omega}^b$ in terms of $\mathbf{q}_{1:3}$ and $\dot{\mathbf{q}}_{1:3}$ from the quaternion kinematics, we get the following:

$$\begin{aligned} \ddot{\mathbf{q}}_{13} &= \begin{bmatrix} 0 & (K_1 + 1)\overline{\omega}_z & (K_1 - 1)\overline{\omega}_y \\ (K_2 - 1)\overline{\omega}_z & 0 & (K_2 + 1)\overline{\omega}_x \\ (K_3 + 1)\overline{\omega}_y & (K_3 - 1)\overline{\omega}_x & 0 \end{bmatrix} \dot{\mathbf{q}}_{13} \\ &+ \begin{bmatrix} K_1(\overline{\omega}_z^2 - \overline{\omega}_y^2) & K_1\overline{\omega}_x\overline{\omega}_y & -K_1\overline{\omega}_x\overline{\omega}_z \\ -K_2\overline{\omega}_x\overline{\omega}_y & K_2(\overline{\omega}_x^2 - \overline{\omega}_z^2) & K_2\overline{\omega}_y\overline{\omega}_z \\ K_3\overline{\omega}_x\overline{\omega}_z & -K_3\overline{\omega}_y\overline{\omega}_z & K_3(\overline{\omega}_y^2 - \overline{\omega}_x^2) \end{bmatrix} \mathbf{q}_{13} + \begin{bmatrix} K_1\overline{\omega}_y\overline{\omega}_z \\ K_2\overline{\omega}_x\overline{\omega}_z \\ K_3\overline{\omega}_x\overline{\omega}_y \end{bmatrix}. \end{aligned}$$

This expression represents a forced linear second-order system whose stability can be checked with the classical theoretical criteria. The previous represents a general case, but we can extract some useful information if we limit ourselves to a simple spin condition around the x axis. Let us assume $\omega_x = \overline{\omega}_x + \delta\omega_x$, $\omega_y = \delta\omega_y$, $\omega_z = \delta\omega_z$ to obtain the following expression:

$$\begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \ddot{\mathbf{q}}_{13} + (I_z + I_y - I_x)\overline{\omega}_x \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \dot{\mathbf{q}}_{13} + \overline{\omega}_x^2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & (I_x - I_z) & 0 \\ 0 & 0 & (I_x - I_y) \end{bmatrix} \mathbf{q}_{13} = 0$$

From this expression, we can already see that around the x axis, there is no dynamical action. Hence, a small perturbation would cause a drift along that axis. More interesting is the stability of the other two axis. We can achieve static stability if the rightmost term is positive; hence, when $I_x >$

Table 5.1 Attitude stability of torque-free motion.

$\omega_x = \bar{\omega}_x + \delta\omega_x,$	$\omega_x = \delta\omega_x,$	$\omega_x = \delta\omega_x,$
$\omega_y = \delta\omega_y,$	$\omega_y = \bar{\omega}_y + \delta\omega_y,$	$\omega_y = \delta\omega_y,$
$\omega_z = \delta\omega_z$	$\omega_z = \delta\omega_z$	$\omega_z = \bar{\omega}_z + \delta\omega_z$
$\ddot{\omega}_x \equiv 0$	$\delta\ddot{\omega}_x \equiv (K_1 K_3 \bar{\omega}_y^2) \delta\omega_x$	$\delta\ddot{\omega}_x \equiv (K_1 K_2 \bar{\omega}_z^2) \delta\omega_x$
$\delta\ddot{\omega}_y \equiv (K_2 K_3 \bar{\omega}_x^2) \delta\omega_y$	$\ddot{\omega}_y \equiv 0$	$\delta\ddot{\omega}_y \equiv (K_1 K_2 \bar{\omega}_z^2) \delta\omega_y$
$\delta\ddot{\omega}_z \equiv (K_2 K_3 \bar{\omega}_x^2) \delta\omega_z$	$\delta\ddot{\omega}_z \equiv (K_1 K_3 \bar{\omega}_y^2) \delta\omega_z$	$\ddot{\omega}_z \equiv 0$
Stable condition: $K_2 K_3 < 0$	Stable condition: $K_1 K_3 < 0$	Stable condition: $K_1 K_2 < 0$

I_z and $I_x > I_y$ (i.e., $K_2 < 0$, $K_3 > 0$). This is only possible if I_x is the maximum moment of inertia of the system.

The previous result does not count all the possibilities for attitude stability as one could restrict to only the second and third directions (i.e., y and z) and compute the characteristic equation of order 4 to find dynamical stability:

$$\det \left\{ \begin{bmatrix} I_y s^2 + \bar{\omega}_x^2 (I_x - I_z) & -(I_z + I_y - I_x) \bar{\omega}_x s (I_z + I_y - I_x) \bar{\omega}_x I_z s^2 \\ + \bar{\omega}_x^2 (I_x - I_y) \end{bmatrix} \right\} = 0$$

$$I_y I_z s^4 + s^2 \bar{\omega}_x^2 (I_x^2 + 2I_y I_z - I_x I_z - I_y I_x) + (I_x - I_z)(I_x - I_y) \bar{\omega}_x^4 = 0,$$

where s is the auxiliary variable.

From the previous equations of fourth degree, we can have acceptable results for stability only if the following conditions are verified:

- $I_y I_z > 0$
- $(I_x^2 + 2I_y I_z - I_x I_z - I_y I_x) > 0$
- $(I_x - I_z)(I_x - I_y) > 0$
- $(I_x^2 + 2I_y I_z - I_x I_z - I_y I_x)^2 - 4I_y I_z(I_x - I_z)(I_x - I_y) > 0$

The first is always true and the third requires I_x to be the major or minor moment of inertia of the system. The other two conditions show that not all the values and ratios for the moments of inertia are allowed to have stability.

We can rewrite the stability equation in terms of K_2 and K_3 as follows:

$$\left(\frac{s}{\bar{\omega}_x} \right)^4 + (1 - K_2 K_3) \left(\frac{s}{\bar{\omega}_x} \right)^2 - K_2 K_3 = 0$$

Giving the following requirements for stability:

- $K_2 K_3 < 1$
- $K_2 K_3 < 0$
- $(1 + K_2 K_3)^2 > 0$

According to the definition of the inertia matrix, we can also state $K_i < 1$ with $i = 1, 2, 3$. Looking at the second requirement, it is straightforward to see that it indicates the second and fourth quadrant of the K_2, K_3 space, as shown in Fig. 5.3. This encompasses also the first requirement as that would include an area comprised between two rectangular hyperbolas on first and third quadrant. Lastly, the third one restricts the ratios out of the hyperbolas

of the second and fourth quadrant, depicted in Fig. 5.3. However, the latter is not very relevant for practical applications. Recalling the statical stability conclusion taken previously (i.e., $I_x > I_z$ and $I_x > I_y$), we can also identify that condition in terms of K_2 , K_3 : $K_2 < 0$ and $K_3 > 0$, which implies that only the second quadrant is also statically stable.

Dual spin dynamics

Dual spin dynamics describes the motion of a spacecraft containing an internal rotating mass, which can be spun independently from the main body. This is relevant because we might be interested in achieving a stable condition regardless the moment of inertia of the spinning axis and the angular rate of the spacecraft. Moreover, dual spin dynamics introduces the concept of internal rotating devices that set the theoretical foundations for attitude dynamics control with angular momentum exchange actuators, which will be discussed in Chapter 7 – Actuators.

Let's assume a spacecraft containing an internal rotating element that can rotate relatively to the z axis of the principal axes body reference frame. The overall angular momentum of the system becomes:

$$\mathbf{h}^b = \mathbf{I} \boldsymbol{\omega}^b + I_r \boldsymbol{\omega}_r = I_x \boldsymbol{\omega}_x \hat{x} + I_y \boldsymbol{\omega}_y \hat{y} + (I_z \boldsymbol{\omega}_z + I_r \boldsymbol{\omega}_r) \hat{z},$$

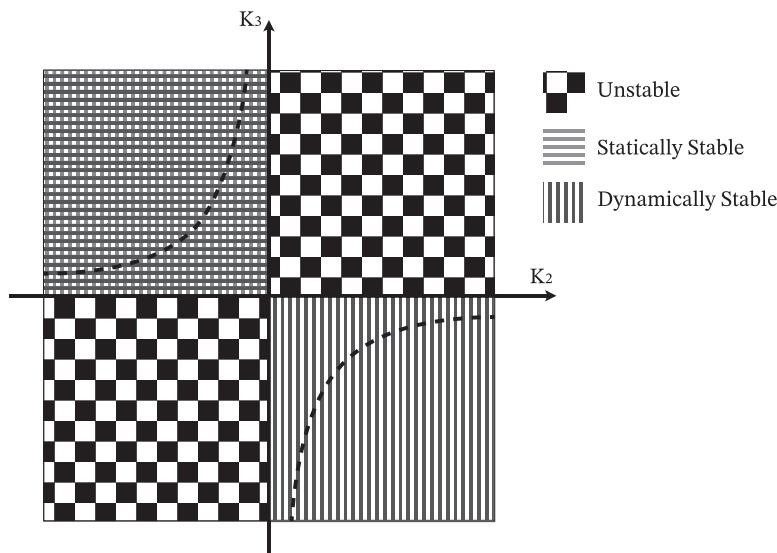


Figure 5.3 Attitude stability graph.

where the moments of inertia of the spacecraft include the inertia of the rotating mass, I_r , and ω_r is the relative velocity of the rotor around the z axis. At this point, we can rewrite the Euler equations in principal inertia axes by using the new definition for the system's angular momentum:

$$\left\{ \begin{array}{l} I_x \dot{\omega}_x + (I_z - I_y) \omega_y \omega_z + I_r \omega_r \omega_y = t_x \\ I_y \dot{\omega}_y + (I_x - I_z) \omega_x \omega_z - I_r \omega_r \omega_x = t_y \\ I_z \dot{\omega}_z + (I_y - I_x) \omega_x \omega_y + I_r \dot{\omega}_r = t_z \\ I_r \dot{\omega}_r = t_r \end{array} \right.,$$

where we added one degree of freedom to the system, and, thus, the fourth equation is required to describe the motion of the internal rotating element, which is driven by the relative torque between the rotor and spacecraft, t_r .

The stability analysis of the dual spin attitude dynamics can be carried out as in the simple spin case, and the interested reader might find the complete derivation in the suggested references [2,4]. However, the main conclusions are relevant for GNC applications. Indeed, dual spin satellites can achieve a stable motion even if rotating around an axis that is not the maximum inertia axis. Generally, if the satellite angular velocity and the rotor angular velocity have the same sign, the ensemble of stable solutions is increased. On the contrary, a counterspinning rotor may reduce the set of potentially stable conditions.

The previous considerations are particularly relevant if the internal rotating mass is operated with a constant angular rate. They can be extended to the case in which the rotor is accelerated to continuously vary its angular rate. In this case, the stability conditions can be defined only for each instant of time, losing their significance. However, this condition is even more relevant for modern GNC, since it allows an active attitude control through the action on the rotating element. In fact, the internal torque applied to the rotor is by reaction transferred to the spacecraft, with an exchange of angular momentum between the main body and the internal mass. As said, this is the fundamental concept behind momentum exchange actuators.

Now it is important to generalize and extend the previous equations to a number of n internal rotating masses aligned with some directions in the body reference frame. The angular momentum can be expressed as:

$$\mathbf{h}^b = \mathbf{I}\boldsymbol{\omega}^b + \mathbf{h}_{\text{rotors}}^b = \mathbf{I}\boldsymbol{\omega}^b + \mathbf{R} \mathbf{h}_{\text{rotors}}^r,$$

where \mathbf{R} is a $3 \times n$ matrix, whose columns represent the direction of the axis of rotation of the internal rotors in the body frame, and $\mathbf{h}_{\text{rotors}}^r$ is a column vector with n elements representing the angular momentum of each rotor around its spin axis. Note that the moment of inertia tensor of the spacecraft includes the inertia of the internal rotating elements. Hence, $\mathbf{h}_{\text{rotors}}^r$ and $\mathbf{h}_{\text{rotors}}^b$ represent the relative angular momentum of the rotors with respect to the spacecraft body. It is relevant to note that $\mathbf{h}_{\text{rotors}}^r$ and $\mathbf{h}_{\text{rotors}}^b$ may not have the same magnitude because \mathbf{R} is generally not an orthogonal matrix. The expression of the matrix \mathbf{R} as a function of the configuration of the internal rotating masses, and the way in which it is used for attitude control will be presented in [Chapter 7 – Actuators](#).

At this point, we can specialize the Euler equation to the case of n internal rotating elements as:

$$\dot{\boldsymbol{\omega}}^b = \mathbf{I}^{-1} [\mathbf{t}^b - \mathbf{t}_{\text{rotors}}^b - \boldsymbol{\omega}^b \times (\mathbf{I} \boldsymbol{\omega}^b + \mathbf{h}_{\text{rotors}}^b)],$$

where $\mathbf{t}_{\text{rotors}}^b$ is the torque applied to the rotors and it has the negative sign because of the Newton's action–reaction principle.

Environmental torques

Satellites orbiting in space are subjected to external disturbances due to the space environment, as it has been discussed in [Chapter 3 – The Space Environment](#). In this section, the effects of the most relevant environmental perturbations are specialized for spacecraft attitude dynamics. Indeed, the environmental torques influencing the Euler equation are described. The section is focused on Earth-orbiting spacecraft, but the discussion can be easily generalized to any other environmental condition.

Gravity gradient torque

Gravity gradient is an environmental torque due to a gravitational field, which is present whenever we have a body that is nonsymmetrical. This torque is commonly the largest perturbation contribution for any spacecraft orbiting in the vicinity of a massive celestial object. Since this torque applies to all bodies characterized by mass with finite dimensions, it also affects planets, moons, and other celestial bodies as well. It is in fact this kind of action that keeps some moons tidally locked to their orbiting planet.

The gravity gradient does not depend on the irregularities in the nonspherical gravitational field, since it is only caused by the fact that the gravitational attraction affects any mass particle of an extended rigid body.

In the case, the body is not symmetric, the summation of the gravity forces over the entire body results in a torque about the center of mass. The actual gravity gradient contribution would depend on the real nonspherical gravity field, but it is usually adequate to approximate the gravity field as spherically symmetric for computing gravity gradient torques in GNC applications. Hence, we will derive the gravity gradient torque on a satellite orbiting only one attractor that is characterized by a spherical and symmetrical mass distribution. The satellite is composed by a single rigid body, and its size is considerably smaller than the distance from the primary center of mass.

Let's divide the spacecraft body into infinitesimal mass elements, dm , which are individually subject to the gravitational attraction of the central planet. The induced torque with respect to the center of mass of the spacecraft can be computed from an integral sum over all infinitesimal masses:

$$\mathbf{t}_g^i = -\mu \int_M \mathbf{r} \times \frac{(\mathbf{r}_{gc} + \mathbf{r})}{\|\mathbf{r}_{gc} + \mathbf{r}\|^3} dm$$

where we have evaluated the torque cross product for any infinitesimal mass in position \mathbf{r} with respect to center of mass of the spacecraft, which has a position vector \mathbf{r}_{gc} relative to the central attractor. The reference frame for the integration has the origin in the center of mass of the body, thus $\int_M \mathbf{r} dm = 0$. Moreover, it should be noted that $\mathbf{r} \times \mathbf{r}_{gc}$ is the only nonnull contribution, and, consequently, the gravity gradient is null on the local vertical (i.e., in the direction parallel to \mathbf{r}_{gc}).

If we expand in a power series the term $\|\mathbf{r}_{gc} + \mathbf{r}\|^3$, considering the assumptions made before (i.e., $\|\mathbf{r}\| \ll \|\mathbf{r}_{gc}\|$), we obtain that the net torque around the center of mass is:

$$\mathbf{t}_g^i = -\frac{3\mu}{\|\mathbf{r}_{gc}\|^5} \mathbf{r}_{gc} \times \left[\left(\int_M \mathbf{r} \otimes \mathbf{r} dm \right) \cdot \mathbf{r}_{gc} \right].$$

Recalling the definition of second moment of inertia, we can substitute the outer product directly and obtain the general:

$$\mathbf{t}_g^i = \frac{3\mu}{\|\mathbf{r}_{gc}\|^3} \frac{\mathbf{r}_{gc}}{\|\mathbf{r}_{gc}\|} \times \left(\mathbf{I} \frac{\mathbf{r}_{gc}}{\|\mathbf{r}_{gc}\|} \right)$$

So, we now have the possibility to define the gravity gradient in the inertial reference frame, given the center of mass position, \mathbf{r}_{gc} , and the inertia

tensor of the body about the center of mass, \mathbf{I} . We can also notice that the magnitude of the gravity gradient torque is inversely proportional to the cube of the distance from the center of the central body, its direction is perpendicular to the radius vector of the center of mass position, and it vanishes if the radius vector is along any principal axis of inertia. Thus, it should be clear that the gravity gradient is an attitude-dependent torque.

It is very convenient to express this torque in the body reference frame. Then, we apply a rotation by the attitude matrix, \mathbf{A} , to \mathbf{t}_g^i and we can express torque in the body fixed frame. If we define $\mathbf{c} = \mathbf{A} \frac{\mathbf{r}_{gc}}{\|\mathbf{r}_{gc}\|}$, we get:

$$\mathbf{t}_g^b = \frac{3\mu}{\|\mathbf{r}_{gc}\|^3} \mathbf{c} \times (\mathbf{I} \mathbf{c}).$$

The vector \mathbf{c} contains the direction cosines of the local vertical direction as seen in the body reference frame, and it can be seen as part of the DCM that links local-vertical-local-horizontal (LVLH) frame with respect to the inertial one. Indeed, \mathbf{c} can be also computed by rotating the x-axis of the LVLH in the body reference frame. The previous expression can be further expanded if we choose the principal axes as the body reference frame and, in this case, we obtain:

$$\mathbf{t}_g^b = \frac{3\mu}{\|\mathbf{r}_{gc}\|^3} \begin{bmatrix} (I_z - I_y)c_y c_z \\ (I_x - I_z)c_x c_z \\ (I_y - I_x)c_x c_y \end{bmatrix}.$$

From this useful expression, we see that a perfectly spherical body would not be affected by the gravity gradient. Tidally locked moons are more oblate and ellipsoidal rather than spherical, and this contributes to keeping them with a stable attitude with respect to the main attractor.

The gravity gradient can enhance or reduce the stability of the system. Let's assume an LVLH moving frame, for simplicity in a circular orbit, and let the x body axis be aligned with the radial zenith direction (i.e., yaw axis), and the z axis parallel to the orbital angular momentum (i.e., pitch axis). Then, we can limit to the first-order expansion getting for small angles around x , y , and z :

$$\mathbf{t}_g^b = \frac{3\mu}{\|\mathbf{r}_{gc}\|^3} \begin{bmatrix} 0 \\ (I_x - I_z) \delta\alpha_y \\ -(I_y - I_x) \delta\alpha_z \end{bmatrix}.$$

Then, assuming an angular velocity with nominal component on the z axis, we would have that for the third equation of motion:

$$\delta\ddot{\alpha}_z = \frac{3\mu}{\|\mathbf{r}_{gc}\|^3} K_3 \delta\alpha_z$$

In this case, if we have $K_3 < 0$, the system would be bounded in this direction as well. This result can be generalized stating that the equilibrium at zero roll and pitch is a stable equilibrium if the x axis (i.e., yaw axis) is the smallest principal moment. Hence, the gravity gradient torque will tend to align a spacecraft with its principal axis of minimum inertia aligned with the radial zenith (or nadir) vector, and it is sometimes used for passive stabilization of spacecraft attitude dynamics. Note that this torque does not depend on the yaw angle. Further insights in gravitational stability can be found in Ref. [4].

Magnetic torque

If a satellite is orbiting a planetary body with a permanent magnetic field, \mathbf{B}^b , and it has a nonzero magnetic dipole moment, \mathbf{m}_m^b , this induces a net torque on the satellite as follows:

$$\mathbf{t}_m^b = \mathbf{m}_m^b \times \mathbf{B}^b$$

where we have expressed all the quantities in the body reference frame. Magnetic moments can arise due to electronics equipment on board the satellite, parasitic currents, and magnetization of ferromagnetic materials as well. In general, the fundamental source of a magnetic dipole is a current loop.

Depending on the level of precision required, the magnetic field can be estimated using a simple dipole model or using a proper magnetic model, as described in the [Chapter 3 – The Space Environment](#). For example, a simplified geomagnetic field can be computed as function of the distance from the Earth center as:

$$\mathbf{B}_{geo} = -\frac{\mu_m}{\|\mathbf{r}_{ge}\|} \begin{bmatrix} 3 \sin \phi_g \cos \phi_g \cos \lambda_g \\ 3 \sin \phi_g \cos \phi_g \sin \lambda_g \\ 3 \sin^2 \phi_g - 1 \end{bmatrix}$$

where ϕ_g is the latitude, λ_g is the longitude, μ_m is the Earth's dipole strength ($\sim 8 \times 10^{22} \text{ A m}^2$), and the resulting magnetic field is expressed in a geomagnetic reference frame, which has to be rotated in body frame to compute the magnetic perturbation torque.

Aerodynamic torque

Atmospheres are constituted by gaseous elements that bound a planetary body, and their interactions with a moving body can be described by fluid dynamics. For orbiting satellites, the atmosphere usually has a low influence, and at relatively high altitudes, it has actually no effect. When a satellite lies in a low orbit or, in general, passes through the upper parts of the atmosphere, it is influenced by the surrounding gas particles. The main effect of the spacecraft–gas interaction is atmospheric drag, since lift is typically almost null, except for very particular missions. The drag can slow down a satellite up to deorbit it, and, in some cases, it can be exploited to actively control the orbits.

For what concern attitude dynamics, aerodynamic torques arise whenever a body has a relative motion with respect to a fluid, even a rarefied one like in the upper atmosphere. A net influence on the attitude is possible if the center of aerodynamic pressure has a relative position with respect to the spacecraft's center of mass, \mathbf{r}_{cpa} . In this way, the drag produces a torque that perturbs the rotational motion:

$$\mathbf{t}_d^b = \mathbf{r}_{cpa} \times \mathbf{f}_d^b.$$

The aerodynamic drag force can be computed as:

$$\mathbf{f}_d^b = -\frac{1}{2} C_d \rho S \| \mathbf{v} \| \mathbf{v},$$

where we used the air density, ρ , which is the main function of the altitude, a drag coefficient, C_d , the area of the surface exposed to the fluid, S , and the relative velocity between the spacecraft and the atmosphere, \mathbf{v} . The aerodynamic drag torque is computed from the drag force vector, considering the position vector of the center of pressure in the spacecraft body frame. The drag coefficient value is commonly set equal to ~ 2 , which is the value

for a flat plate. In fact, the aerodynamic drag torque is practically computed by summing up all the drag contributions of each spacecraft flat surface. Very simple applications may assume the spacecraft as composed by a single flat surface, while more complex models may use a polyhedral discretization of the spacecraft shape. Experimental analyses are needed if the atmospheric drag shall be estimated with high accuracy.

The major characteristic of aerodynamic drag is that it is always opposed to the velocity with respect to the atmosphere. Hence, a proper relative velocity computation with respect to the atmosphere of a rotating body must be performed and not be neglected [1]. Further details about atmospheric drag and atmosphere models are discussed in the [Chapter 3 – The Space Environment](#).

Solar radiation pressure torque

Pretty much like for atmospheric drag, spacecraft attitude is also perturbed by the luminous radiation coming from the Sun. Light can behave like a wave or like a particle, and when these particles interact with bodies having specific optical properties, a momentum transfer can occur. The entity of the force of the sunlight hitting a body is extremely low, but it remains in effect for every satellite that is not in shadow. In low Earth orbit, the effect of solar radiation pressure (SRP) is dominated by aerodynamics, but SRP torques will generally prevail over aerodynamic torques in higher altitude orbits. Like for the atmospheric case, the resulting force is computed with an integral over the surfaces exposed to the Sun radiation and depending on the application point with respect to the center of mass, we could have a net torque on the satellite.

If we analyze the incoming radiation on a surface, we can exploit the particle interpretation of light to deal with these particles hitting the surface. The infinitesimal force $d\mathbf{f}_s^b$ is a function of the momentum exchange and it can be determined as:

$$\left\{ \begin{array}{l} d\mathbf{f}_s^b = -p \cdot \xi(\alpha) \cos \alpha \hat{\mathbf{s}} \, dA \\ p_{srp} = \frac{SF}{c} \\ \cos \alpha = \hat{\mathbf{s}} \cdot \hat{\mathbf{n}}_A \\ \xi(\alpha) = \frac{1}{2} (\text{sign}(\cos \alpha) + 1) \end{array} \right.$$

where we have determined the force per unit area as function of the intensity of the pressure p , the direction opposite to the one of the traveling photons $\hat{\mathbf{s}}$ (i.e., satellite-Sun direction), and an impact angle given by the dot product with the normal direction of the infinitesimal surface $\hat{\mathbf{n}}_A$. Note that the overall SRP is computed integrating the system of equations over the spacecraft's surfaces. It should be noted that the coefficient $\xi(\alpha)$ enforces the fact that if a surface is not lighted up (i.e., $\cos \alpha < 0$), the net force should be zero.

The underlying assumption of the previous equations was that all momentum is transferred to the body (i.e., absorbed radiation), which is rather idealistic and not representative of the reality of electromagnetic radiation hitting a surface. In fact, depending on the material of the surface, we could experience a total absorption, a total transmission, a specular reflection, or, more generally, a diffuse reflection. Thus, not all the incoming radiation is absorbed, but the overall incoming radiation is divided into fractions that experience the four aforementioned possibilities:

$$\sigma_{\text{absorb}} + \sigma_{\text{diffuse}} + \sigma_{\text{specular}} + \sigma_{\text{transmit}} = 1,$$

and obviously they shall sum up to one.

Only absorption and reflection contribute to the overall momentum transmission. Thus, with the current assumptions, the infinitesimal SRP force on a generic elemental surface of the spacecraft is expressed as:

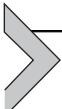
$$d\mathbf{f}_s^b = - p \cdot \xi(\alpha) \cos \alpha \left((\sigma_{\text{absorb}} + \sigma_{\text{diffuse}}) \hat{\mathbf{s}} + \left(\frac{2}{3} \sigma_{\text{diffuse}} + 2 \sigma_{\text{specular}} \cos \alpha \right) \hat{\mathbf{n}}_A \right) dA$$

The net force is computed through an integral over the considered exposed area, and the net torque is given by the cross product with the center of luminous pressure location in body reference frame:

$$\begin{cases} \mathbf{t}_s^b = \mathbf{r}_{c_{ps}} \times \mathbf{f}_s^b \\ \mathbf{r}_{c_{ps}} = \frac{\int \xi(\alpha) \cos \alpha \mathbf{r} dA}{\int \xi(\alpha) \cos \alpha dA}, \end{cases}$$

It should be noted that the surface has been assumed to be convex or flat, as concavities might complicate the formulation and would request more analysis and self-shadowing considerations. Note that reflected light from the planetary bodies, called albedo, can be significant if very precise

dynamical modeling is required. Further details about SRP are discussed in the [Chapter 3 – The Space Environment](#).



Three-body problem attitude dynamics

Whenever the spacecraft is subject to a 3BP dynamical environment, the attitude motion is influenced by both the primary attractors, as in the case of the orbital motion described in the [Chapter 4 – Orbital Dynamics](#). The main peculiarity of the 3BP attitude dynamics is the presence of the two-gravity gradient torques that deeply characterize the attitude motion in the 3BP. Attitude dynamics (i.e., coupled orbit–attitude dynamics) in three-body environment is getting an increased attention for GNC applications, since modern space missions will be often set in 3BP environments, such as the Cislunar space, or the Sun–Earth system.

Euler equations are formulated to include the gravity torques exerted by the two primary attractors, which can be computed as the gravity gradient torque defined in the section about environmental torques. Moreover, the equations of attitude motion may further include the most relevant perturbations in the orbital environment. For example, SRP torque and the gravity gradient torques of other celestial bodies are commonly included to accurately model 3BP attitude dynamics [6]. The resulting Euler dynamical equations for the attitude dynamics in three-body environment are expressed as:

$$\left\{ \begin{array}{l} I_x \dot{\omega}_x + (I_z - I_y) \omega_y \omega_z = (I_z - I_y) \left[\frac{3(1-\mu)}{\|\mathbf{r} - \mathbf{r}_1\|^3} l_y l_z + \frac{3\mu}{\|\mathbf{r} - \mathbf{r}_2\|^3} h_y h_z \right] + t_x \\ I_y \dot{\omega}_y + (I_x - I_z) \omega_x \omega_z = (I_x - I_z) \left[\frac{3(1-\mu)}{\|\mathbf{r} - \mathbf{r}_1\|^3} l_x l_z + \frac{3\mu}{\|\mathbf{r} - \mathbf{r}_2\|^3} h_x h_z \right] + t_y \\ I_z \dot{\omega}_z + (I_y - I_x) \omega_x \omega_y = (I_y - I_x) \left[\frac{3(1-\mu)}{\|\mathbf{r} - \mathbf{r}_1\|^3} l_x l_y + \frac{3\mu}{\|\mathbf{r} - \mathbf{r}_2\|^3} h_x h_y \right] + t_z \end{array} \right.$$

where l_i are the direction cosines in the body reference frame of the unit position vector from the first primary to the spacecraft; h_i are those related with the unit position vector from the second primary to the spacecraft; t_i are other external torques acting on the spacecraft. The other quantities were introduced in the [Chapter 4—Orbital Dynamics](#): \mathbf{r} refers to the spacecraft

body position vector; \mathbf{r}_1 and \mathbf{r}_2 are the primaries position from the three-body system's barycenter; and μ is the mass ratio of the 3BP primaries.

Attitude dynamics in 3BP allows to investigate periodic orbit–attitude motions in three-body environments and to analyze the orbit–attitude coupling in these dynamical regimes. Further details can be found in dedicated reference studies [6–11].



Relative attitude dynamics

Analogously to the case of translational dynamics, relative attitude dynamics refers to the description of the attitude motion of one body with respect to a rotating reference frame. Such relative dynamics is quite common in missions involving rendezvous and docking, formation-flying, on-orbit servicing, and proximity operations, which are common applications in modern spacecraft GNC.

Let's assume a spacecraft whose body frame is indicated with C , and another reference frame that can be attached to a different orbiting spacecraft, or to a fictitious point following a trajectory in space. This second reference frame is indicated with T . Then, we define the attitude kinematics in terms of quaternions, \mathbf{q}_C and \mathbf{q}_T , relating the inertial reference frame with C and T . The relative attitude kinematics is defined by exploiting the relative quaternion definition as:

$$\delta\mathbf{q} = \mathbf{q}_C \times \mathbf{q}_T^{-1} = \begin{Bmatrix} \chi(\mathbf{q}_T) \mathbf{q}_C \\ \mathbf{q}_T^T \mathbf{q}_C \end{Bmatrix},$$

where $\chi(\mathbf{q}_T)$ is the 3×4 matrix defined in the relative quaternion kinematics section. Note that the relative quaternion kinematics follows the same rules of the kinematics of absolute quaternions.

The rotation matrix \mathbf{P} that transforms a vector from the reference frame T to the frame C can be expressed in terms of the relative quaternion $\delta\mathbf{q}$ as:

$$\mathbf{P}(\delta\mathbf{q}) = \begin{bmatrix} 1 - 2(\delta q_2^2 + \delta q_3^2) & 2(\delta q_1 \delta q_2 - \delta q_3 \delta q_4) & 2(\delta q_1 \delta q_3 + \delta q_2 \delta q_4) \\ 2(\delta q_1 \delta q_2 + \delta q_3 \delta q_4) & 1 - 2(\delta q_1^2 + \delta q_3^2) & 2(\delta q_2 \delta q_3 - \delta q_1 \delta q_4) \\ 2(\delta q_1 \delta q_3 - \delta q_2 \delta q_4) & 2(\delta q_2 \delta q_3 + \delta q_1 \delta q_4) & 1 - 2(\delta q_1^2 + \delta q_2^2) \end{bmatrix}.$$

The relative angular velocity between the two reference frames can be expressed in the frame C as:

$$\delta\omega^C = \omega^C - \mathbf{P} \omega^T,$$

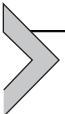
where ω^C and ω^T are the angular rates of the two reference frames C and T expressed in the respective reference frames.

At this point, the relative attitude dynamics of the reference frame C with respect to the reference frame T can be expressed in C as:

$$\begin{aligned}\delta\dot{\omega}^C = & \mathbf{I}_C^{-1} \left\{ - [\delta\omega_x^C] \mathbf{I}_C \delta\omega^C - [\delta\omega_x^C] \mathbf{I}_C \mathbf{P} \omega^T + \mathbf{I}_C [\delta\omega_x^C] \mathbf{P} \omega^T \right. \\ & - [\mathbf{P} \omega_x^T] \mathbf{I}_C \delta\omega^C + \mathbf{t}^C - \mathbf{P} [(\mathbf{P}^T \mathbf{I}_C \mathbf{P} - \mathbf{I}_T) \mathbf{I}_T^{-1} (\mathbf{t}^T \\ & \left. - [\omega_x^T] \mathbf{I}_T \omega^T) + [\omega_x^T] (\mathbf{P}^T \mathbf{I}_C \mathbf{P} - \mathbf{I}_T) \omega^T] - \mathbf{P} \mathbf{t}^T \right\},\end{aligned}$$

where \mathbf{I}_C and \mathbf{I}_T are the inertia tensors of the two reference frames in principal axes; \mathbf{t}^C and \mathbf{t}^T are the external torque vectors acting on the rigid bodies, respectively, expressed in C and T .

Relative attitude dynamics can be used in relative GNC applications to derive GNC functions directly under a relative perspective. Its formulation is very useful in modern spacecraft missions, which are more and more directed toward relative and close proximity applications, formations flying, rendezvous, and docking. Further considerations may be found in specific reference studies [12,13].



Multibody spacecraft dynamics

For most applications, a satellite can be modeled as a rigid single body. However, there are modern applications where this is no longer possible. If the satellite has long flexible components (e.g., solar panels and radiators) or has movable appendages (e.g., robotic arms, booms, or antennas), the assumption is not valid anymore. In some cases, it is possible to estimate a model of the forces and torques on the flexible and movable elements to be applied at interfaces with the main body. This simplified approach is possible for moving elements and flexible appendages in the case of a very massive base (e.g., space station). If these secondary masses are not so different from the main body ones, the effects of appendages and flexible elements can be quite important, and the spacecraft has to be modeled with a multibody approach.

One common movable appendage is a robotic arm installed on the spacecraft body. For one robotic arm only made by revolute joints, one can refer to Refs. [14,15]. In this case, the center of mass is a key factor to describe all the positions of the single elements. In fact, a multibody satellite

has the center of mass and center of rotation still coincident, meaning that position and attitude of the base depend on the rest of the elements as well. Another modeling approach is to use a Euler–Lagrange formulation that can be applied to a single chain of rigid bodies. In this case, the exchanged forces are sequentially applied in one direction and then resolved in the opposite direction to estimate all the relevant forcing contributions. A further approach consists in generalizing the concept of bodies chained together in multiple branches and enforcing the center of mass and rotation properties. More complex cases with closed networks of bodies may require further investigations and the possible usage of a more general multibody solver [16,17]. It should be noted that the robotic arm might affect only a small portion of a satellite mission: no satellite should be moving around with a robotic element fully operative if it is not the time to use it.

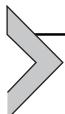
Flexible appendages usually require a finite element model analysis to estimate frequencies. Then, this information is integrated as an external torque applied to the system, as described in the [Chapter 3 – The Space Environment](#). However, if the flexible elements need be closely integrated with the spacecraft model, this is still possible with a multibody approach. In this case, the flexible appendages are modeled as discretized elements, linked to the main body with joints, dampers, and springs, inducing equivalent forces and flexible dynamics in the rigid system. The discretization level influences the representativity of the flexible dynamics, and it is directly responsible of the computational load to simulate the system behavior.

Other interesting sources of disturbances for attitude dynamics are the mass expulsion or aggregation events. The most common is the loss of mass due to the expulsion of material through the propulsive subsystem. This is often modeled as an equivalent force, and the mass expulsion is rarely described with a multibody approach. However, this can be done to accurately represent the time history of the center of mass' position. In fact, the propellant temporal variation can be combined with the propellant and fluid sloshing inside the satellite. These terms can be modeled through a series of masses and multibody elements that can mimic the sloshing frequencies and the propellant expulsion.

In general, attitude dynamics is strictly related with movable parts, flexible elements, and internal fluids. Thus, accurate modeling of attitude dynamics requires dedicated multibody flexible models of the system. The reader is invited to deepen this topic on the suggested references and to find further details in the [Chapter 3 – The Space Environment](#).

References

- [1] F.L. Markley, J.L. Crassidis, Fundamentals of Spacecraft Attitude Determination and Control, Space Technology Library, Springer, New York, 2014.
- [2] P.C. Hughes, Spacecraft Attitude Dynamics, Dover Publications Inc., New York, 2004.
- [3] V.A. Chobotov, Spacecraft Attitude Dynamics and Control, Orbit Book Co, 1991.
- [4] J.R. Wertz, Spacecraft Attitude Determination and Control, vol. 73, Springer Science & Business Media, New York, 1978.
- [5] M.H. Kaplan, Modern Spacecraft Dynamics and Control, Wiley, New York, 1976.
- [6] A. Colagrossi, M. Lavagna, Preliminary results on the dynamics of large and flexible space structures in Halo orbits, *Acta Astronautica* 134 (2017) 355–367, <https://doi.org/10.1016/j.actaastro.2017.02.020>.
- [7] A. Colagrossi, M. Lavagna, Dynamical analysis of rendezvous and docking with very large space infrastructures in non-keplerian orbits, *CEAS Space Journal* 10 (1) (2018) 87–99, <https://doi.org/10.1007/s12567-017-0174-4>.
- [8] D. Guzzetti, K.C. Howell, Natural periodic orbit-attitude behaviors for rigid bodies in three-body periodic orbits, *Acta Astronautica* 130 (1) (2017) 97–113, <https://doi.org/10.1016/j.actaastro.2016.06.025>.
- [9] D. Guzzetti, K.C. Howell, Attitude dynamics in the circular restricted three-body problem, *Astrodynamicas* 2 (2) (2018) 87–119, <https://doi.org/10.1007/s42064-017-0012-7>.
- [10] F. Colombi, A. Colagrossi, M. Lavagna, Floquet modes and stability analysis of periodic orbit-attitude solutions along Earth–Moon halo orbits, *Celestial Mechanics and Dynamical Astronomy* 133 (34) (2021), <https://doi.org/10.1007/s10569-021-10030-y>.
- [11] A. Colagrossi, V. Pesce, L. Bucci, et al., Guidance, navigation and control for 6DOF rendezvous in Cislunar multi-body environment, *Aerospace Science and Technology* 114 (2021) 106751, <https://doi.org/10.1016/j.ast.2021.106751>.
- [12] G.Q. Xing, S.A. Parvez, Alternate forms of relative attitude kinematics and dynamics equations, in: 2001 Flight Mechanics Symposium, Greenbelt, Maryland, USA, 19–21 June, 2001, pp. 83–97. <https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/2001084965.pdf>.
- [13] S.-G. Kim, et al., Kalman filtering for relative spacecraft attitude and position estimation, *Journal of Guidance, Control, and Dynamics* 30 (1) (2007) 133–143.
- [14] K. Yoshida, B. Wilcox, Space Robots, Springer handbook of robotics, 2008, pp. 1031–1063.
- [15] K. Yoshida, Achievements in space robotics, *IEEE Robotics and Automation Magazine* 16 (4) (2009) 20–28.
- [16] A.A. Shabana, Dynamics of Multibody Systems, Cambridge university press, 2003.
- [17] A.A. Shabana, Flexible multibody dynamics: review of past and recent developments, *Multibody System Dynamics* 1 (2) (1997) 189–222.



Sensors

**Andrea Colagrossi¹, Vincenzo Pesce², Stefano Silvestrini¹,
David Gonzalez-Arjona³, Pablo Hermosin⁴, Matteo Battilana⁵**

¹Politecnico di Milano, Milan, Italy

²Airbus D&S Advanced Studies, Toulouse, France

³GMV Aerospace & Defence, Madrid, Spain

⁴Deimos Space, Madrid, Spain

⁵OHB Italia S.p.A., Milan, Italy

Sensors on-board a spacecraft are the source for any information, measurement, or data used by the guidance, navigation, and control (GNC) system to accomplish its task. In analogy with our human experience, they are the sensory organs (e.g., eyes, ears, nose, etc.) sensing the external environment to acquire information needed for the GNC operations. A proper GNC design shall select and arrange the sensors on the spacecraft system in a way that all the desired quantities are properly measured. In doing so, the GNC designer shall know, understand, and correctly model the sensors to support its design process. This chapter is dedicated to briefly present all the major sensor typologies for spacecraft GNC applications, to discuss their operating principles, to highlight their peculiarities, to categorize the available components according to their performances, and to discuss the most relevant modeling techniques and rules.

The content of this chapter is structured as follows:

- *Sensor modeling for GNC.* In this section, the sensor modeling techniques are presented, discussing the main implementation principles of the most common numerical models to include the sensor behavior in the GNC design, analysis, verification, and validation phases. The main error sources and their modeling are introduced, together with the most typical sensor faults. Moreover, this section contains some elements of metrology, and it explains some basic principles of statistics and random variables representation.
- *Orbit sensors.* This section presents the Global Navigation Satellite System (GNSS) and the ground-based sensors for orbit GNC. Their main operating principles and the available performance are discussed. A simple GNSS sensor model implementation in MATLAB/Simulink is included in this section.

- *Attitude sensors.* This section introduces the main typologies of sensors for attitude GNC. For each of them, it presents the main characteristics and the operational principles. The available performance of the different attitude sensors is compared, and an example Sun sensor model implemented in MATLAB/Simulink is also described.
- *Inertial sensors.* Accelerometers and gyroscopes are presented in this section. Their peculiar error sources are listed and presented, together with the available performance according to the quality grade of the sensor. A brief introduction to the Allan variance (AVAR) representation of their random errors is included. The chapter is concluded with a simple example of a MATLAB/Simulink implementation of a gyroscope model.
- *Electro-optical sensors.* This section introduces the sensors with a vision-based sensing capacity, such as cameras and Light Detection and Ranging (LIDAR) systems. The two typologies, respectively, represent the class of passive and active electro-optical sensors. Their applicability range and their design principles are briefly discussed.
- *Altimeters.* In this section, the concept of altimetry is introduced and the most relevant typologies of altimeters for spacecraft applications are discussed. A mathematical altimeter model is also presented at the end of this section.



Sensor modeling for GNC

Correct sensor modeling is of utmost importance for GNC applications and design. Indeed, algorithms development and testing shall rely on proper sensor models to have a truthful representation of the available measurements. Moreover, some estimation filters are based on measurement models that are capable to include the terms to be estimated. Thus, we need to understand what the model of a sensor is, how it is built, and how it is used along the GNC design and verification processes.

A sensor is an electronical device exploiting a physical principle to measure a certain physical quantity, such as the angular velocity, the position of the spacecraft, or the position of a celestial body, among the others. The external reality is sensed by the sensor's detector, elaborated by the sensor processing units, and communicated to the GNC system via electrical signals, which can be analogic or digital, even if modern spacecraft applications are almost uniquely based on digital components. Along this entire process, the measured physical quantity is inevitably distorted and modified with

respect to the reality, and a good sensor model shall be able to represent these deviations, together with all the effects due to the sensing activities. We are going to mathematically refer to this as:

$$\check{\mathbf{x}}(t) = \mathbf{x}(t) + \boldsymbol{\epsilon}(t), \quad (6.1)$$

where $\check{\mathbf{x}}(t)$ is the measured quantity, $\mathbf{x}(t)$ is the real-world true quantity, and $\boldsymbol{\epsilon}(t)$ is the summation of all the sensing process errors and effects.

A sensor model is an abstraction of the actual sensing process converting the physical quantity into the electrical output, and it is implemented for a precise purpose with specific objectives. In fact, there exist different types of sensor models: operational, behavioral, functional, and performance models are the most common.

Operational sensor models are used to describe and represent how the sensor is operated and interfaced with the rest of the GNC system, in terms of commands, operational modes, and interfaces with other components. Behavioral models reproduce the global behavior of the sensor, including transitions between modes, response to inputs, and evolution of the outputs. However, behavioral models do not model the internal processes of the sensor, but they describe an averaged transient or steady-state behavior. Functional sensor models are established to serve as input for detailed analyses and accurate performance simulations. They outline all the internal functions executed along the sensor processing, being representative of the sensor actual temporal performances for realistic output profiles. The complete equipment acquisition chain is typically modeled, from signal transduction and conditioning to digitalization and formatting. Functional models can be representative of real input and output data formats, but they are not required to include different operational modes. Consequently, more functional models may be necessary to simulate the system in its entire operational domain, with all its different modes. Performance sensor models represent the way in which the sensor operates and interacts with the GNC system, in terms of performances, consumed resources, and physical limitations. Thus, they are not focused on the operations, functions, and behaviors of the sensor, but on the implications the performance of the sensor has on the surrounding system. They are used to verify how the GNC system works, highlighting potential bottlenecks of the design, given the performance of the selected components. Performance models use equivalent generic mathematical models, and they are commonly modeling only the principal and the most influent characteristics of the sensors.

Functional and performance sensor models are the most used for GNC design, analysis, simulation, and verification. In fact, they respectively allow to assess the detailed functionalities and the overall performance of any element of the system. Moreover, specific analyses and verification procedures may require a combination of the features of the aforementioned sensor models. For example, an operational-functional sensor model can be used to verify with great accuracy a GNC subsystem in diverse operative modes. In addition, there are very specific and application-oriented sensor models for telemetry and telecommand (TMTC), i.e., models to generate sensor telemetry packets or react to telecommands, or for failure mode, effects, and criticality analysis (FMECA) analyses, i.e., models to simulate faults or effects due to failures.

Sensor model's interfaces need to be defined according to the specific type of model in use and considering the desired application. The interfaces of the sensor model determine the input/output of the system simulator, which is used for GNC design and verification, and their fidelity level shall be selected accordingly. In fact, the sensor model's interfaces may be formed by representative engineering data or by the raw sensor data. The former are numeric values, whose data format is not relevant for the specific application, and they are directly used by the algorithms in the mathematical formulas. The latter are representative of the raw values, as communicated by the real sensor hardware with a specific data format. Thus, they need to be processed and converted in the desired format for the following operations. For this reason, raw sensor data interfaces are typically used when the complete GNC verification is of interest, with particular focus on the verification of GNC software components. Alternatively, engineering data are handy when the GNC design is on-going, and the analyses are focused more on the GNC algorithms performance than on the final development and implementation.

Sensor models are exploited both in accelerated and in real-time simulators. Accelerated simulations typically support the early development and verification stages, while the real-time models are integrated in dedicated testing benches to verify the latest hardware and software developments, as will be extensively discussed in [Chapter 12](#) — GNC Verification and Validation. It is convenient to implement a model having in mind both applications, since it is very common to port the model from the accelerated simulation environment (e.g., MATLAB/Simulink) to the real-time one. In these regards, the requirements for the real-time model (e.g., compatibility with auto-coding techniques, compatibility with real-time

target and execution) should be considered since the beginning of the development in the accelerated simulation environment. The two typologies of simulated models shall be validated together to confirm that they produce the same results, and they have the same temporal evolution. In particular, the execution order of the accelerated simulation shall be carefully checked to ensure coherence with the real-time application. This last point may be facilitated if the sensor model is developed using atomic unit blocks [1] in accordance with the real sensor processing steps.

Like any other part of a GNC system, the sensor models shall be validated, meaning that they shall be truly representative of the real sensor component. The validation campaign shall be focused both on the short-time and on the long-time evolution of the sensing process errors and effects, noting that the long-time ones have a deeper impact on the GNC performance. Ideally, the validation should also address the variation of the sensor behavior with the external environment (e.g., temperature, pressure, acceleration state). A sensor model is typically developed exploiting the data available in the datasheets and in the documentation provided by the producer. However, in the later stages of the GNC development and verification, the model parameters must be updated in accordance with real measured sensor data. Note that the data obtained from the exact flight model of the sensor installed on the spacecraft should be used for a thorough and comprehensive validation campaign.

The primary use of the sensor models is the support of GNC design and verification emulating the presence of the sensors to be installed on-board. However, the sensor models are also directly used in the GNC blocks. For instance, guidance functions to calibrate or initialize sensors, navigation filters measurement models, control methods to reduce effects disturbing a particular sensor rely on a correct modeling of the sensor measurements. Hence, as will be discussed in [Part 2 – Spacecraft GNC](#), it is common to use dedicated sensor models or specific sensor variables directly inside the GNC functions.

Elements of metrology

Sensors convert the physical quantity to measure into an electrical signal following a predefined sequence of transduction and processing steps, which

is denoted as sensor acquisition chain. It depends on the actual implementation of the specific sensor, but it is generally composed by:

- Transduction.
- Conditioning.
- Analog-to-digital (AD) conversion.
- Calibration and transmission.

Transduction is the process of converting the physical property into the corresponding electrical signal. Conditioning is needed to process the transduced sensor signals into a form that can be converted to digital values, and it is typically composed by signal filtering, amplification and shaping, and linearization. AD conversion allows the digitalization of the conditioned electrical signals by converting the analog input voltage or current to a digital number representing the magnitude of the voltage or current. Then, the digital signal can be calibrated to convert back the electrical data into the equivalent physical quantity that has been measured. Finally, the calibrated digital signals are encoded and formatted to reduce transmission errors and to guarantee compatibility with the communication protocol of the sensor.

[Fig. 6.1](#) summarizes a generic sensor acquisition chain. Note that sensors for space applications commonly transmit their data by exploiting typical communication interfaces, such as Universal [Synchronous and] Asynchronous Receiver-Transmitter (U[S]ART), Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I2C), Controller Area Network (CAN), etc.

Sensors can measure absolute quantities or differential ones. In the first case, which is the most common for spacecraft applications, the sensor is designed to seek the “true” value of the physical quantity to measure. While, in the case of differential sensors, the measurement is performed to seek a deviation with respect to a “reference” value. This is done to minimize systematic errors, which are affecting both the reference and the differential measurements. Note that in absolute sensors, the measurement error is the deviation between the obtained measurement result and the true value, while differential sensors have the errors affecting the difference between

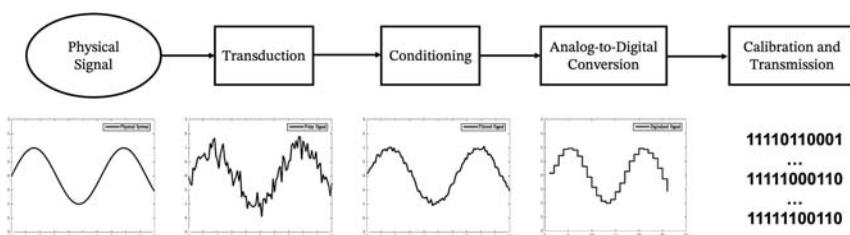


Figure 6.1 Sensor acquisition chain.

the reference and actual measurements. As a result, good differential measurements can be obtained even with a system of poor quality. Anyhow, the following discussion will be focused on absolute sensors.

Measurement errors, $\epsilon(t)$, are impossible to be exactly known, but they can be assessed under a statistical perspective. Thus, we shall introduce the concept of measurement uncertainty, which gives us information about the amount of measurement error that is likely to occur. The measurement uncertainty is defined as a “nonnegative parameter characterizing the dispersion of the quantity values being attributed to the measured physical quantity” [2]. In fact, the measurement result is a random variable, which is described by a probability density function that is typically bell-shaped (i.e., normal statistical distribution). An ensemble of more measurement results is centered at a certain statistical average value and dispersed according to a statistical standard deviation (or variance). These two quantities are associated to two concepts about sensors: accuracy and precision.

The output of the sensor is the measure, $\hat{x}(t)$, which is, by definition, different from the real measured quantity. The closer the measure is to the real world, the more accurate the sensor is. Indeed, the accuracy of a sensor is how close the measurement is with respect to truth (i.e., the distance of the statistical average value with respect to the true value). The precision of a sensor is the reproducibility with which a measurement is made (i.e., the dispersion of the normal distribution, related to the statistical standard deviation). We shall not confuse the definitions of these two concepts, visually represented in Fig. 6.2. Note that for each realization of the measurement, there is a single resulting error value, which relates each measure to the actual value according to Eq. (6.1). Hence, the statistical quantities behind accuracy and precision combine themselves identifying the interval containing the measurement result values.

In general, the main concepts characterizing the sensors are:

- *Accuracy.* The mean difference that will exist between the actual value and the indicated value at the output of the sensor.
- *Precision.* The degree of reproducibility of a measurement, which is associated to closeness between measured quantity values obtained by replicate measurements on the same quantity.
- *Sensitivity.* The minimum input of the physical parameter that will create a detectable output change, which is associated to the slope of the input-to-output characteristic curve (i.e., output change for a given change in the input).

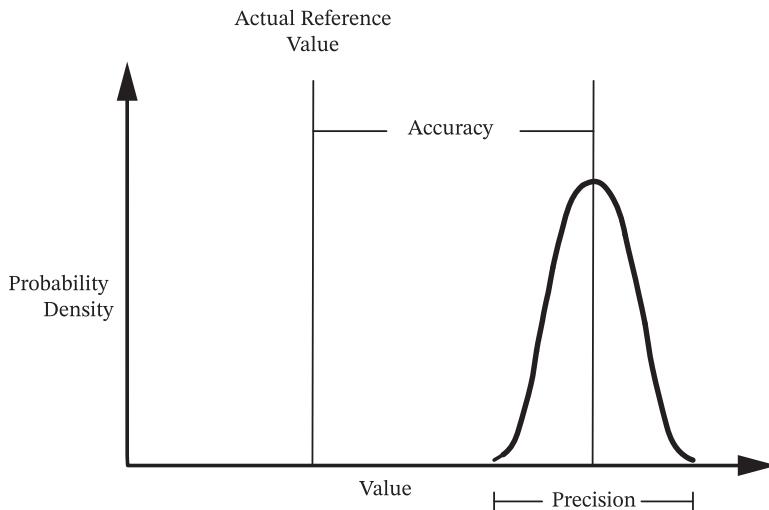


Figure 6.2 Sensor accuracy and precision.

- *Range*. The maximum and minimum values of the applied parameter that will be measured.
- *Resolution*. The smallest detectable incremental change of input parameter that will be detected in the output signal.
- *Offset (bias)*. The output that will exist when it should be zero.
- *Linearity*. The deviation the actual measured curve of a sensor will have from the ideal input-to-output linear characteristic curve.
- *Hysteresis*. The difference the actual measured curve of a sensor will have as a function of the input variation direction.
- *Response time*. The time required for a sensor output to change from its previous state to a final settled value within a tolerance band of the correct new value.
- *Dynamic linearity*. The ability of a sensor to follow rapid changes in the input physical parameter.

All the previous concepts characterize the behavior of the sensor, and they shall be included in a proper sensor model.

Inferring from the previous discussion, we understand how precision is a description of random errors (e.g., noise, instabilities) and accuracy is a description of systematic errors (e.g., offset, imperfect calibrations). Before entering the details of both error typologies, we shall focus on the fundamentals of probability and stochastic processes, and on the systematic errors induced by a wrong calibration.

Probability and stochastic processes

Random errors require a statistical interpretation to be understood, and the fundamental concept behind this is the probability. By definition, the probability of an event A to occur is defined by how likely this event is to happen and it can be generally defined as:

$$P(A) = \frac{\text{number of times } A \text{ occurs}}{\text{total number of occurrences}}.$$

However, the probability of a given event A to occur can be also linked to an event B that happened before. In this case, we talk about conditional probability of A given B and we can define it as:

$$P(A|B) = \frac{P(A, B)}{P(B)}$$

where $P(A, B)$ is the joint probability of A and B and it describes the probability that both A and B occur.

Random variables

Another very important concept to introduce when dealing with the statistical interpretation of random errors is the one of random variable. A random variable is a mathematical description of the outcome of random events. In a broader sense, it can be seen as a variable mapping a set of random events to a set of real values. The most interesting thing about random variables is that they have associated properties very useful in the domain of probability and statistics. In particular, one of the most important quantities associated to a random variable is the probability distribution function (PDF), defined as:

$$F_X(x) = P(X \leq x)$$

with, in this case, X being a generic random variable and x a nonrandom variable or constant. Similarly, the probability density function (pdf) can be defined as:

$$f_X(x) = \frac{dF_X(x)}{dx}.$$

Also for random variables, conditional quantities can be defined, given that a certain event A has occurred:

$$F_X(x|A) = P(X \leq x|A),$$

$$f_X(x|A) = \frac{dF_X(x|A)}{dx}.$$

In order to quantify the average value of a given random variable over a large number of experiments, it is necessary to introduce the concept of expected value. The expected value of a random variable X , or mean of X , is defined as:

$$E(X) = \mu = \frac{1}{N} \sum_{i=1}^m A_i n_i = \int_{-\infty}^{\infty} x f_X(x) dx$$

where N is the number of total experiments, m is the total number of possible different outcomes, and A_i is a given event occurring n_i times. Please recall that $f_X(x)$ is the pdf of X . Using the definition of expected value, it is possible to also define the concept of variance. The variance of a random variable is a measure of how a given random variable will vary from its expected value or mean. The variance of a given random variable X is defined as:

$$\sigma_X^2 = E[(X - \mu)^2] = \int_{-\infty}^{\infty} (x - \mu)^2 f_X(x) dx$$

where σ_X is also known as standard deviation of a random variable.

In general, a random variable X can be defined as $X \sim (\mu, \sigma^2)$, having mean μ and variance σ^2 .

Different types of random variables and PDFs exist and have been categorized in statistical literature [3]. Here, we will recall the most important ones for spacecraft GNC applications.

Uniform random variables

A uniform random variable has, by definition, a pdf that is constant between two limits a, b . In practical terms, the random variable X has an equal probability of assuming any value in the interval defined by the pdf limits and zero probability outside the interval. In a mathematical way, it is defined as:

$$f_X(x) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

A graphical representation of a uniform distribution is shown in Fig. 6.3.

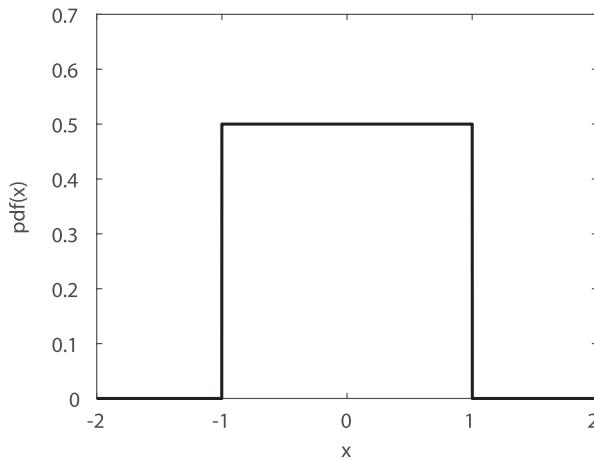


Figure 6.3 Uniform distribution.

Gaussian random variables

A random variable is defined Gaussian or normal if its pdf is expressed as:

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left[\frac{-(x-\mu)^2}{2\sigma^2}\right]}$$

The graphical representation of a zero-mean Gaussian normal distribution with unitary variance is reported in Fig. 6.4.

An important consideration about Gaussian random variables is that, with this distribution, the variable will have 68% of probability to assume a value:

$$X = \mu \pm 1\sigma$$

In other words, there will be 68% odds for the variable to lie within one standard deviation from the mean, 95% of odds to lie within two standard deviations from the mean, etc. It is a normal practice to define the error as being specified at 1σ , or, more conservatively, at 3σ , or whichever level. Recalling Fig. 6.2, it can also be interpreted as the likelihood for the error to assume that value. As an applicative example, if the designer models the random error as a constant noise with its 3σ value, it means that it will overestimate the actual error 99.7% of the times.

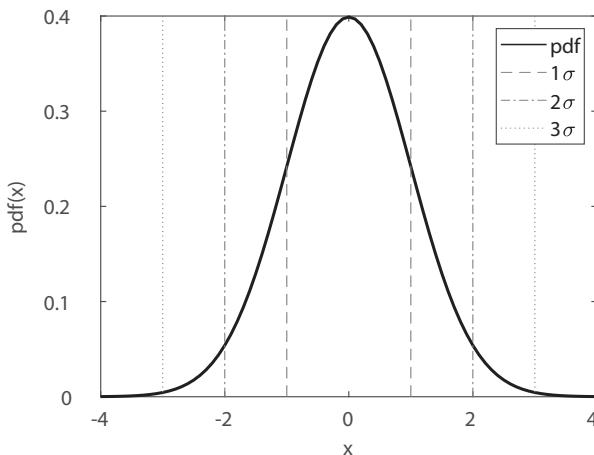


Figure 6.4 Gaussian or normal distribution.

Stochastic processes

The variation of a random variable in time can be represented through a stochastic process. A stochastic process $X(t)$ describes the variation in time of a random variable X . As previously done for a random variable, PDF, pdf and expected value of a stochastic process can be defined. In particular, the PDF takes the form of:

$$F_X(x, t) = P(X(t) \leq x)$$

and for the pdf:

$$f_X(x, t) = \frac{dF_X(x, t)}{dx}.$$

Similarly:

$$\begin{aligned} E(X(t)) &= \mu = \int_{-\infty}^{\infty} xf_X(x, t) dx \\ \sigma_X^2 &= E[(X - \mu)(X - \mu)^T] = \int_{-\infty}^{\infty} (x - \mu)(x - \mu)^T f_X(x, t) dx \end{aligned}$$

While talking about stochastic processes, if a random variable at a given time t_1 is not dependent on its value at another instant of time t_2 , we call this stochastic process as white noise. Otherwise, we talk about colored noise,

with different types of colors depending on the stochastic process power spectrum. These concepts will be deepened in the following of this chapter, when discussing about sensor noise and random errors. Moreover, the reader is invited to extend its knowledge on statistical methods in literature references [3,4] and in the appendix [Chapter 16 – Mathematical and Geometrical Rules](#).

Sensor calibration

Sensor calibration is crucial for GNC applications, since a good calibration campaign can minimize most of the systematic errors, improving the system's accuracy up to the best nominal accuracy of the selected sensors. It should be remarked that the usage of accurate sensors without a proper calibration is a huge mistake under the engineering perspective and a waste of resources under the programmatic perspective. Sensor calibration is the comparison of measurement values delivered by the sensor with those of a calibration reference standard. Strictly speaking the calibration definition does not include the adjustments to improve the measurement accuracy, but it is commonly implied that the calibrating adjustments are performed along the calibration procedures. Spacecraft sensors are calibrated to remove statistical biases introduced by mounting errors, axis misalignments, temperature dependencies, geometrical distortions, environmental influences, scale factors, nonlinearities, and other potential sources of systematic errors. A complete calibration can be very complex, and a trade-off shall be performed between achievable accuracy and calibration effort. Note that a perfect calibration is practically not achievable because of the intrinsic finite accuracy of the calibration reference standard and the finite precision of the sensor. Moreover, it is suggested to have a calibration effort that is coherent and balanced with the precision of the sensor, the overall accuracy of the other sensors on the spacecraft, and the required accuracy of the GNC system.

The calibration shall be performed on-ground during the GNC development, especially for those errors related with the assembly and integration phases (e.g., misalignments, distortions). However, any calibration is not perpetual, and successive recalibrations shall be performed at prescribed intervals of time. If this is very common and standardized for Earth applications, the same is not true for spacecraft due to the physical inaccessibility of the platform. So, we should integrate calibration standards on-board or use the known space environment — or the spacecraft dynamics — as

calibrating references. In any case, on-board sensor calibration is suggested for those applications looking for a high accuracy. It can be performed at discrete intervals, or it can be continuously executed. Navigation and estimation functions are capable to quantify biases on-board to recalibrate the sensors, as will be discussed in [Chapter 9](#) – Navigation and in the on-board sensor calibration of [Chapter 14](#) – Applicative GNC Cases and Examples.

Errors modeling

Sensor models for GNC applications are based on mathematical models of the processes along the sensor acquisition chain. They take as input a simulated physical signal, and they are set to output the sensor measurements. The format and the characteristics of the output depend on the typology of sensor model, which also influences the way in which the internal processes are modeled. As said, the most complete functional models include almost every step of the acquisition chain, as shown for a generic functional sensor model in [Fig. 6.5](#). Note that other typologies of sensor models may be composed by equivalent subsets of the blocks used for a functional model.

With reference to [Fig. 6.5](#), the most relevant systematic and random error sources for spacecraft sensors are:

- Bias or zero offset errors.
- Scale factor errors.
- Noise or random fluctuations errors.
- Quantization or resolution errors.
- Misalignment and nonorthogonality errors.

Moreover, other sensor processing phenomena exist, and even if they cannot be defined as errors, they shall be included in proper sensor models:

- Time delays and latencies.
- Maximum range or output saturation.
- Output temporal discretization.

The sensor acquisition chain is also influenced by the environment (e.g., temperature, acceleration, radiations, etc.). The environmental influences

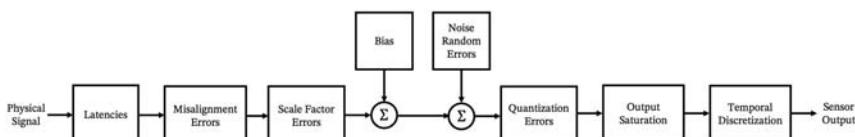


Figure 6.5 Generic functional sensor model example.

are not immediately evident from the previous discussion, since the changes in the external environmental conditions affect the sensor output in an indirect way. That is, the previously listed error terms are dependent from the external environment. For example, the bias and the scale factor errors may be dependent from the accelerations, the noise may vary as a function of the temperature, or other similar dependencies. In general, the cross talk between potentially interacting physical quantities (e.g., acceleration, magnetic field, angular velocity, temperature, etc.) shall be carefully investigated and accounted during the implementation of the sensor models.

Sensor models may be also influenced by additional aspects, such as aging, contamination from pollutant, light source reflections, and occultations. Thus, also these effects shall be modeled and considered, if relevant for the specific application.

Finally, note that sensors whose processing is dependent from the time (e.g., GNSS sensors) may be subject to indirect errors because of clock inaccuracies or time deviations with respect to the universal timing.

The most relevant error terms are briefly described in the followings, whose modeling numerical parameters are commonly found in the datasheet of the sensor.

Bias

Measurement bias is a systematic error that set an offset in the output signal with respect to the actual physical value. Typically, bias randomly varies at each start-up of the sensor, and it is influenced by the external environment. It is assumed to be constant or slowly varying in time, and its temporal fluctuations are categorized within the random error sources. It is one of the error sources most prone to be minimized, thanks to calibration.

Its mathematical model is:

$$\check{\mathbf{u}} = \mathbf{u} + \mathbf{b},$$

where **b** is randomly selected for each simulation around the nominal value stated in the sensor's datasheet.

Scale factor errors

Scale factor errors are a class of systematic errors that are proportional to the magnitude of the input signal. Scale factor errors may be of linear, nonlinear, or asymmetric type. Linear scale factors (or just scale factors) introduce a nonunitary slope coefficient between output and input. Nonlinear scale

factors are proportional to the square, or higher order terms, of the input, and they let the output curve deviate from a straight line. Asymmetry scale factors are proportional to the absolute value of the input, and they create an angle in the output curve close to the zero value. Scale factors may be influenced by the external environment, especially temperature, and should be modeled in the entire operative range of the sensor. Calibration techniques are effective in reducing scale factor errors.

Scale factor models can be mathematically described as:

$$\check{\mathbf{u}} = \mathbf{u} + \mathbf{SF}\mathbf{u} + \mathbf{SF}_N\mathbf{u}^2 + \mathbf{SF}_A|\mathbf{u}|,$$

where \mathbf{SF} , \mathbf{SF}_N , and \mathbf{SF}_A , are, respectively, the linear scale factor, nonlinear scale factor, and asymmetry scale factor coefficients. These coefficients are usually expressed in parts per million, PPM. Note that 1 PPM corresponds to $10^{-6} = 0.0001\%$ of the input value.

Noise and random errors

Random errors are nonsystematic errors that influence any sensor, since they are intrinsically related to the stochastic nature existing in real-world applications. Because of their fundamental property, they are impossible to be fully compensated and a residual stochastic measurement error is present in any measure. In fact, the only way to reduce the effects of random errors is to average the measure from a large set of measurements. This is not always possible, since this procedure introduces a delay in the measurement processing that may be not acceptable for the specific GNC application. We will refer to noise alternatively to random error or stochastic process. In general, noise is used as a generic term to indicate any undesired errors generated by a stochastic process.

Random errors shall be described in terms of their statistical parameters, since a deterministic description is ontologically not possible. They can be of various typologies, but they are always characterized by an unpredictable and not replicable time evolution. Thus, they are associated to a standard deviation, σ , or variance, σ^2 , but they commonly have null average because constant errors are taken into account together with bias errors. The probability distribution of random measurement errors is typically well approximated by a Gaussian normal distribution since, according to the central limit theorem, they are the result of the sum of many independent random variables. This can also be seen in Fig. 6.6, where a noise signal with $\sigma = 3$ is sampled for 100 s at 100 Hz. From the figure it can be seen how the noise signal samples

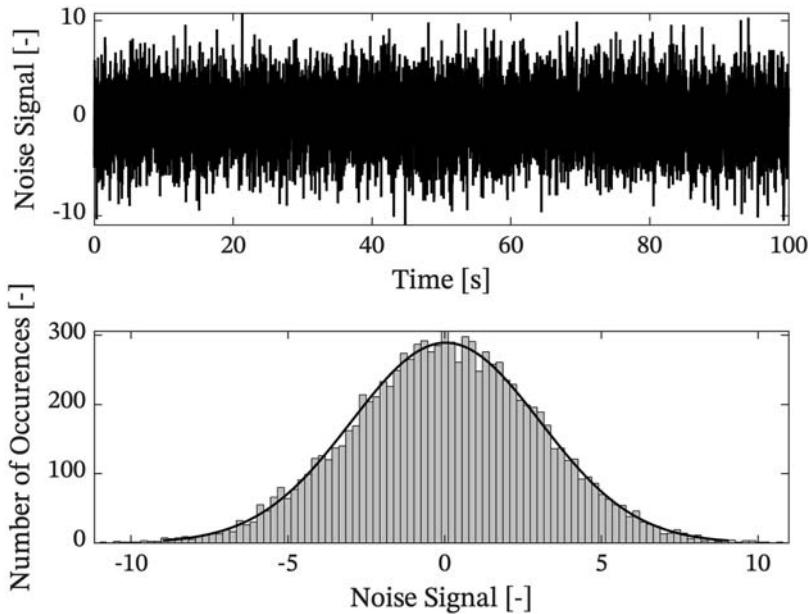


Figure 6.6 Noise signal with $\sigma = 3$.

are well fitted by a normal distribution with the same standard deviation. As a result, $\sim 68\%$ of the samples are within the σ interval (i.e., $[-3, 3]$), while $\sim 99.7\%$ of them are contained in the 3σ interval (i.e., $[-9, +9]$). Indeed, almost all the samples have an absolute value lower than 10.

Random errors are also associated with different frequencies of the stochastic process behind them, and their resulting output can be generally identified by analyzing their power spectrum. A signal whose spectrum has equal power (i.e., intensity) within any equal interval of frequencies is defined as white noise; its power spectral density is constant. White noise is the most common noise source in sensor measurements, since it is usually introduced by the electronic sensor processing. However, there exist other random errors with a frequency spectrum that is not constant. They are defined as colored noises, and according to the shape of their power spectral density, they are associated with a color. For instance, pink noise has a power spectrum which decreases linearly in the logarithmic scale, while blue noise increases. Other colors may be defined, but they cannot be generalized. The most common noises that are relevant in sensor modeling are:

- White noise.
- Pink noise, or flicker noise, power spectral density decreases proportionally to $1/f$, where f is the frequency.
- Brown noise, or Brownian noise, or red noise or also identified as random walk noise power spectral density decreases proportionally to $1/f^2$, and it is the noise associated with the random walk errors affecting many digital sensors. Brownian noise can be generated with temporal integration of white noise.

An example of these noise sources with $\sigma = 10$ is reported in Fig. 6.7, in terms of their signal sampled for 1000 s at 100 Hz, and their power spectral density.

Noise modeling shall exploit random number generators from a normal distribution, defining the standard deviation and a parameter, usually denoted as seed, to initialize the random generator. Note that seed values shall be different between different components in the model, and there should be the possibility to change their value at each simulation to avoid simulating always an identical scenario. The plain normal random generator can model a white noise source. Filtering the random number generator's output can be used to simulate colored noise. For example, a low-pass filter with a power roll-off of 3 dB per octave (i.e., the signal power reduces by a factor of 2 every time the frequency doubles) can be applied to a random number generator to model a pink noise. Similarly, a low-pass filter with a power roll-off of 6 dB per octave can be used to model a brown noise.

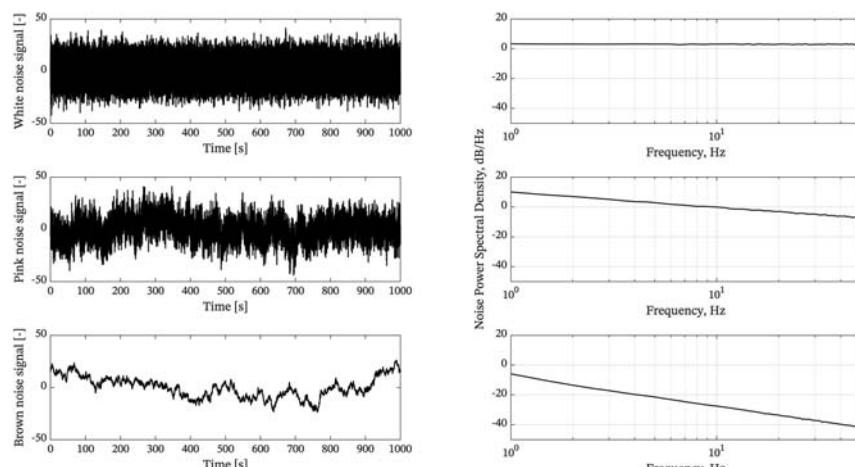


Figure 6.7 White, pink, and brown noise and their power spectral densities.

The low-pass filter should be set with a cut-off frequency at -3 dB as low as possible in order to be applied on the entire frequency spectrum. The cut-off frequency may be slightly varied to precisely tune the noise frequency content. Note that brown noise can be also modeled with a time integrator placed at the output of the random number generator. This is commonly done to model bias instabilities and fluctuations in scale factor errors. Example noise models are shown in Fig. 6.8.

Noise amplitude is tuned by varying the standard deviation in the random number generator. However, sensor datasheets do not frequently report this value, and the fine tuning of the sensor model shall be supported by sensor characterization testing activities. This is especially true for colored noise terms. Noise density is more often reported on the datasheets. In fact, the actual noise amplitude frequently depends on the sampling time of the sensor. Let's assume the noise amplitude, A , expressed in the units of the measured quantity, u , then the standard deviation of the normal random numbers shall be:

$$\sigma = A u.$$

The noise amplitude density, D , is measured in units over the square root of the frequency, f (i.e., $u/\sqrt{\text{Hz}}$), and it can be related to the amplitude as:

$$A u = D \sqrt{\frac{f}{2}} = \frac{D}{\sqrt{2T_s}},$$

where T_s is the sampling time of the sensor.

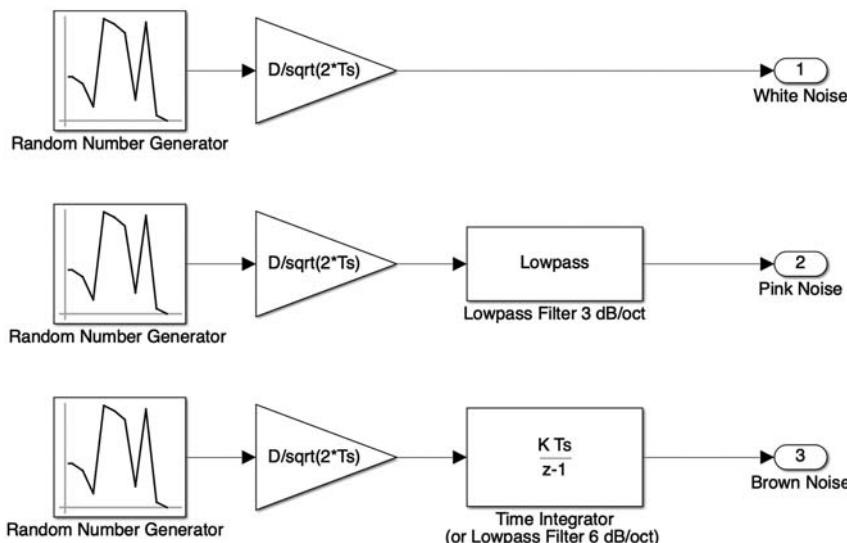


Figure 6.8 Noise models.

Random errors with uniform distribution

The Gaussian normal distribution is not the one for all PDF, since stochastic processes may be distributed according to different PDFs. As said, Gaussian distribution is important for engineering applications because of the central limit theorem, which states that, when measuring real-world data, the different measured values tend to produce a final distribution that is approximately normal. Generally, the more a measurement is like the sum of independent variables with equal influence on the result, the more normality it exhibits. Indeed, we can often regard a single measured real-world data value as the weighted average of many small independent effects.

However, there are situations in which this condition does not hold, and the random errors shall not be modeled by exploiting the normal distribution. Binomial distribution, Poisson distribution, uniform distribution, exponential distribution, and the Weibull distribution are examples of alternative PDFs. The former two are often used to represent the statistical distribution of discrete data, while the latter applies to continuous valued data [4]. They are often used in various specialized applications, but the most relevant among them for spacecraft GNC is the uniform distribution.

The uniform distribution was introduced before in the random variables section, and it is used to represent those data in which every value within a certain interval has the same probability to occur. For instance, the position of the center of mass during the design process of a spacecraft will be typically assumed to fall inside a box with uniform distribution, since any point within the acceptability range has the same probability to occur. Similarly, the uncertainties on the values of other geometrical or inertia properties of the spacecraft are commonly represented with a uniform distribution. In the same way, if the GNC design considers an initial condition for the spacecraft attitude dynamics after the deployment from the launcher, these are typically uniform in terms of the attitude states. In fact, all the rotations of the body frame with respect to the inertial one are possible.

In general, for any random variable, the GNC designer shall select the proper PDF. As a rule of thumb, measured real-world data are normally distributed, while the uncertainties in the system parameters or the unknown initial conditions are uniformly distributed within the acceptability ranges.

Quantization errors

Quantization error is a systematic error resulting from the difference between the continuous input value and its quantized output, and it is like

round-off and truncation errors. This error is intrinsically associated with the AD conversion that maps the input values from a continuous set to the output values in a countable set, often with a finite number of elements. The quantization error is linked to the resolution of the sensor. Namely, a high-resolution sensor has a small quantization error. Indeed, the maximum quantization error is smaller than the resolution interval of the output, which is associated to the least significant bit representing the smallest variation that can be represented digitally:

$$LSB = \frac{FSR}{2^{N_{BIT}}}$$

where FSR is the full-scale range of the sensor, and N_{BIT} is the number of bits (i.e., the resolution) used in the AD converter to represent the sensor's output. Quantization errors are typically not corrected, and the discrete values of the output are directly elaborated by the GNC system, which is designed to operate on digital values.

[Fig. 6.9](#) shows a convenient model block to simulate quantization errors.

Misalignment and nonorthogonality errors

Misalignment and nonorthogonality errors are systematic errors due to mounting errors of the sensor, internal misalignment in the sensor's axes, or nonorthogonality between the different axes of an N-dimensional sensor. These errors can be easily minimized via ground calibration during the testing phase on the integrated spacecraft. However, residual and unknown misalignment and nonorthogonality are anyway present in the sensor measurements. Misalignment makes a measure to be not correctly associated to a nominal reference axis. Nonorthogonality determines a cross-axes sensitivity, since a physical quantity along an axis has nonzero spurious measured components also on the other axes of the sensor. Moreover, the nonorthogonality errors typically account also for the cross-axes electrical interference between the different axes of the sensors.

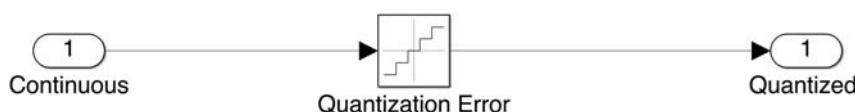


Figure 6.9 Quantization error model.

Misalignment and nonorthogonality errors are mathematically represented by multiplying the physical signal by rotation, \mathbf{R} , and nonorthogonal, \mathbf{O} , matrices, respectively:

$$\check{\mathbf{u}} = \mathbf{O} \mathbf{R} \mathbf{u}.$$

The rotation matrix, \mathbf{R} , is an orthonormal matrix rotating the physical signal into the sensor misaligned axes by a set of random error angles. These misalignment angles are usually very small (i.e., m°), especially if the sensor is carefully calibrated. The nonorthogonal \mathbf{O} matrix may be computed by multiplying a lower-triangular matrix with a symmetric one [5]; they result in a matrix with almost unitary components on the diagonal and small random values on the lower and the upper nonsymmetric triangular blocks. The small random values are in the order of the nonorthogonality or cross-axes errors, which are commonly expressed in percent units (i.e., 5% corresponding to about 0.05 cross-axes contribution).

Output saturation, temporal discretization, and latencies

Sensor's outputs are also characterized by saturation, temporal discretization, and time delay, or latency, with respect to the physical input.

Output saturation is simply related to the maximum dynamic range of the sensor: an input value larger (or smaller) than the maximum (or minimum) range saturates the sensor's output. It can be modeled by including a saturation in the sensor model. Output saturation should not create problems in GNC application, since sensors are usually selected to have a greater dynamic range with respect to the expected dynamic ranges of the measurable quantities.

Temporal discretization of the output is due to the discrete time operations of a real GNC system, which interrogates the sensor at discrete intervals of time. Thus, this phenomenon can be modeled by maintaining a constant output between two discrete time intervals. Temporal discretization frequency of the measurements shall be compatible with the intended operating frequency of the GNC.

Latency is due to the internal processing, and it is typically quantified with ground testing. This effect can be modeled including a delay in the sensor model, and it is particularly relevant for applications with stringent synchronization requirements or with many sensors sampled at different instants of time.

Fig. 6.10 shows common model blocks used to represent these effects.

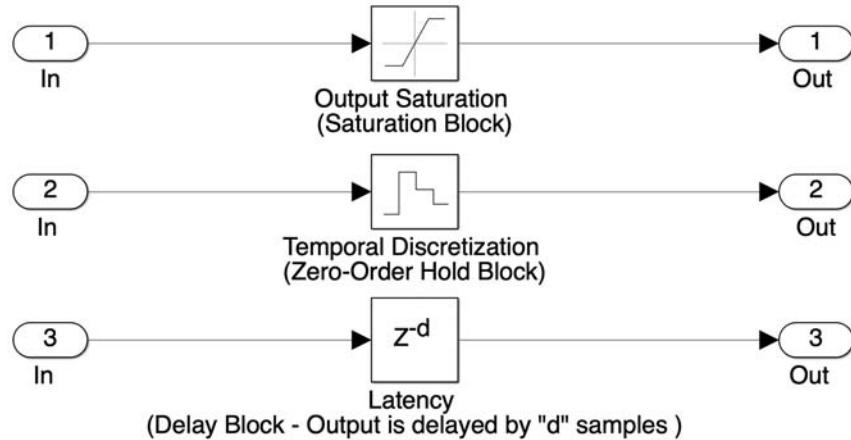


Figure 6.10 Latency, output saturation, and temporal discretization.



Sensor faults

Sensor faults have a critical impact on the GNC system. A faulty sensing unit may introduce wrong measurements in the GNC loop, leading to navigation errors bringing the system off the reference state. Similarly, the loss of a sensor may bring the spacecraft in a critical error state, where the complete functionality of the spacecraft may be not guaranteed. For these reasons, the GNC design shall take into account possible sensor faults, and it shall comply with unexpected on-board failures. Ideally, these undesired events shall be detected by the GNC itself, which shall also react to minimize their impact on the whole spacecraft. These points will be addressed in [Chapter 11 – FDIR Development Approaches in Space Systems](#), but here a brief summary of the most common typologies of sensor faults is reported.

Sensor faults are classified and listed according to their impact on the sensor's output [6,7]. The most common are:

- *Spike fault.* Spikes are observed in the output of the sensor.
- *Erratic fault.* Noise of the sensor output significantly increases above the usual value.
- *Drift fault.* The output of the sensor keeps increasing or decreasing from nominal state.
- *Hardover/bias fault.* The output of the sensor has a sudden increase or decrease with respect to the nominal state.

- *Data loss fault*. The output of the sensor has time gaps when no data are available.
- *Stuck fault*. The sensor's output gets stuck at a fixed value.

[Fig. 6.11](#) reports their typical appearance in a generic sensor's output.

It shall be noted that sensor models shall be also capable to reproduce the possible faults that may occur while on orbit. Hence, sensor faults need to be simulated in order to reproduce some failure scenarios that resembles real-life faults and failures by utilizing the concept of fault injection into the simulated system.

Orbit sensors

Orbit sensors are used to measure position and velocity of a satellite by using different signals and technologies. When position and velocity measurements are available, they can be used to estimate the current orbital state and, possibly, to propagate the spacecraft position forward in time. Orbit

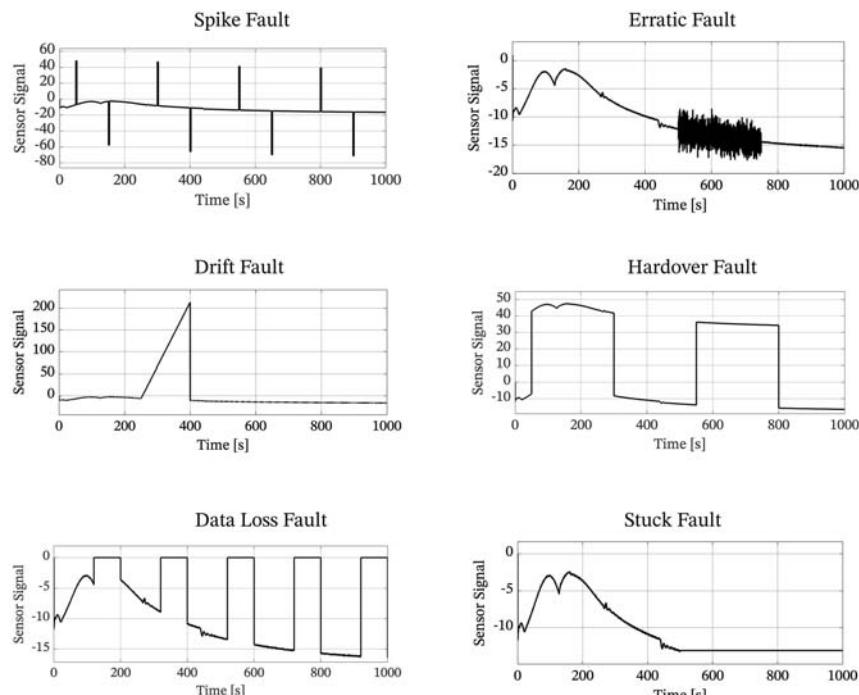


Figure 6.11 Typical sensor faults appearance, adapted from Colagrossi and Lavagna [[7](#)].

sensors belong to two main typologies: on-board or on-ground. The former enables autonomous GNC, since the measurements are directly available on the spacecraft, while the latter requires the ground support, with the upload of the measured position and velocity during the telecommunication windows.

The orbit sensors a given spacecraft can carry strongly depend on the mission and the spacecraft itself. For instance, the use of GNSS receivers is not useful for a Mars mission since GNSS satellites can only provide their signals to receivers below GNSS orbits (except for very weak signals), but it would provide a great solution for orbit determination for low Earth orbit (LEO) satellites. Therefore, when selecting the sensors suite for a given satellite, it is essential to have in mind the operational orbit and characteristics of the spacecraft. As a general consideration, on-board orbit sensors (e.g., GNSS) provide a higher level of autonomy with respect to on-ground sensors at the cost of limited applicability domain. In fact, on-board sensors are usually limited to Earth-orbit scenarios where the lower complexity of the mission and the available technologies allow for precise and autonomous orbit determination. On the contrary, the complexity to enable interplanetary orbit determination usually drives the sensor selection toward on-ground solutions. In fact, they are general and not limited to Earth scenarios, but they add operational costs, delays, and lack in autonomy.

Thus, the different types of orbit sensors can be classified depending on the autonomy and the applicability range of the navigation solutions that can be obtained by using them. In this section, on-board GNSS sensors and on-ground orbit determination with ground station networks (GSNs), along with their corresponding sensor technologies are discussed.

GNSS sensors

GNSS sensors allow a satellite or, generally speaking, any object that is below the GNSS satellites orbits (e.g., a smartphone on Earth surface) to determine its position by using the signals generated by the GNSS constellation satellites next to the ephemerides of those satellites.

GNSS basics

GNSS refers to a constellation of satellites that provides signals to receptors (that can be on land, sea, air, or even in space) with information of position and time for them to being able to determine their location. Taking into account the time it takes to signals to go from the emitting satellites in the constellation to the receptors and knowing their detailed position, it is

possible to obtain a range measurement between the emitter and the receptor. If enough signals are received, the GNSS receptor can estimate the state. There are different sets of constellations that provide global coverage, from which we can remark:

- **GPS [8]:** GPS is the oldest and more known GNSS constellation. It was designed and built, and it is currently operated and maintained by the US Department of Defense. It consists of 24 satellites, spread in 6 different orbital planes with four satellites each. The orbits semimajor axis is 26,578 km, and they have an inclination of 55°.
- **GLONASS [9]:** GLONASS is the GNSS managed by the Russian Space Forces, and it is operated by the Ministry of Defense of the Russian Federation. It consists of 21 satellites divided in 3 planes with seven satellites each. Satellites operate in nearly circular orbits with a semimajor axis of 25,510 km and an inclination angle of 64.8°.
- **GALILEO [10]:** GALILEO is the GNSS satellite constellation from the European Union and the European Space Agency with the objective of providing a very high-accuracy global positioning system under civilian control. The GALILEO constellation consists of 30 satellites in medium Earth orbit (MEO) orbit in three orbital planes with 9 satellites per orbit (plus one spare). Orbits are quasicircular with a semimajor axis of 29,600 km and 56° of inclination.
- **BEIDOU [11]:** BEIDOU is the satellite constellation from China. It consists of 35 different satellites, divided in different planes and orbit regimes. From the 35 satellites in the constellation, 8 are in geostationary regime (five in geostationary orbit [GEO] orbit and other three in inclined GEO) and the other 27 are MEO satellites that allow for complete coverage.

Apart from the global satellite positioning constellation, there are other systems that provide regional coverage in order to increase the accuracy and improve the performance of the global navigation systems, such as EGNOS or QZSS.

EGNOS is a European regional satellite-based augmentation system that uses GNSS measurements taken by accurately located reference stations deployed across Europe to improve the reliability and the accuracy of the GNSS positioning data [10].

QZSS or “Quasi-Zenith Satellite System” is a regional augmentation system for the Asia–Pacific region, especially focused on Japan, which is the developing country of the system. The QZSS system consists of 4

satellites in inclined GEOs that allow a complete regional coverage and enhancement of GNSS [12].

GNSS signals

The main objective of a GNSS satellite is to transmit a signal that is then received by a user and interpreted to extract position and velocity information. The transmitted signal has a frequency in the radio spectrum of about 1.2 and 1.6 GHz ($L1 = 1.575$ GHz, $L2 = 1.243$ GHz, $L5 = 1.176$ GHz) with a peculiar code modulation. In fact, pseudorandom noise code modulation is used on the harmonic radio wave (carrier). As the name suggests, this modulation has no evident pattern and it consists of a defined random sequence of zeros and ones, repeated at a given time interval and it serves as ranging code. Furthermore, a broadcast navigation message is transmitted along with the ranging code but at a lower rate. This navigation message provides information on the orbit and clocks of the transmitting satellites. More information can be found in Ref. [13].

The receiving user can interpret the GNSS signal data and extract three types of measurements:

- Pseudorange: given the time of the receiver clock at signal reception t_R and the time of satellite clock at signal transmission t_T , their difference scaled by the speed of light c can be used to construct a range measurement:

$$P = c(t_R - t_T)c$$

- Carrier phase: a direct measurement of the signal phase can be used as more precise measurement. First, the code modulation has to be removed to recover raw phase, then, the phase should be tracked to record the number of cycles. Phase information is usually more precise than pseudorange, but its initial value generates an integer ambiguity N_R . If the phase tracking is lost, the carrier phase measurements exhibit a cycle slip. A simple measurement equation considering the phase of the receiver Φ_R and the phase of the transmitted signal Φ_T is:

$$\Phi = \Phi_R(t_R) - \Phi_T(t_R) + N_R^T$$

- Doppler: the well-known Doppler effect causes a change in the received frequency. A direct measure of this change provides information on the line-of-sight velocity.

In this chapter, a simple introduction to GNSS signals is provided. For additional details on the GNSS measurements modeling, including all the relevant error effects, please refer to [Chapter 9](#) – Navigation.

GNSS receivers

A GNSS receiver is typically mounted in LEO satellites to perform orbit determination. The GNSS receiver can provide position, velocity, and timing data (among others, such as pseudorange and carrier phase information depending on the specific receiver) that allows tracking the satellite state along the orbit. Satellites in other orbital regimes such as MEO or GEO can make use of GNSS receivers [14], and there have been different projects to study the possibility of using GNSS signals to perform navigation around the Moon, even though the precision is not as high as the one for receivers in LEO orbit and below [15].

The GNSS receivers incorporate an antenna to get the signal provided by the different GPS satellites. In principle, having three satellites in view would be enough for the receiver to determine the exact position in three-dimensional (3D) space, see [Fig. 6.12](#). The received signals include

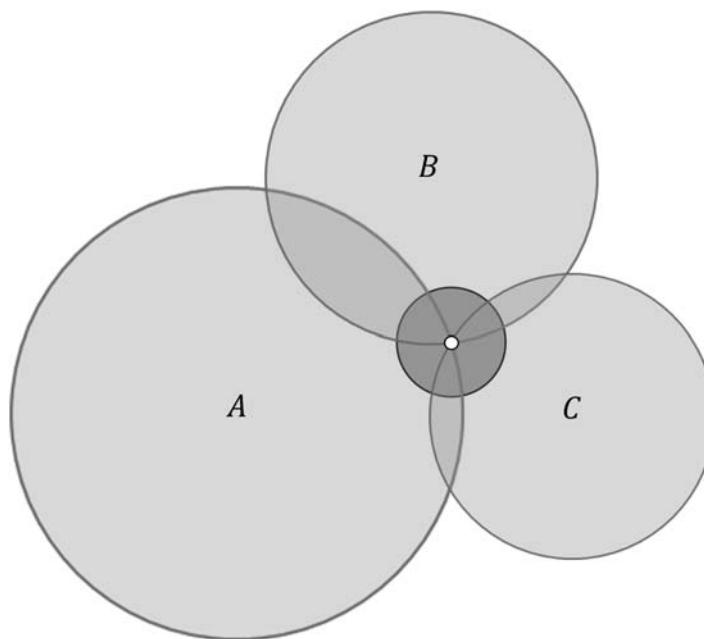


Figure 6.12 Triangulation scheme from three different satellites.

the exact timestamp of when they were generated. Therefore, by considering the generation time and the reception one, it is possible to determine the distance between the GNSS satellite that transmitted the signal and the receiver. If at least three satellites are in view of the receiver, by means of triangulation, it is possible to determine the position of the satellite with a few meters error. In modern GNSS navigation, a minimum of four satellites is usually considered to perform orbit determination.

It is important to mention that, in order for a spacecraft to be able to determine the distance with respect to the GNSS satellite, it is mandatory to know the position of the GNSS satellite itself. A set of ground stations is constantly monitoring and obtaining orbit determination solutions of the GNSS satellites. These orbital data are stored by using numerical parameters, and they are available on the GNSS satellites in the GNSS ephemerides and almanacs data. The ephemerides are broadcasted by the GNSS satellites in near real time, and they are updated at a frequency of few hours. The ephemerides allow the receiver to determine its position with respect to the satellite of the constellation in view.

However, there is an extra factor to be taken into account when determining the position from GNSS satellites: clock errors. GNSS satellites are equipped with high precision atomic clocks to maintain tracking of time, but receivers cannot be equipped with these types of clocks, since it would make them extremely expensive. For this reason, it is necessary to have a fourth satellite in view of the receiver to solve for the extra variable, time, and provide with an accurate solution of the localization problem.

GNSS accuracy

The accuracy of GNSS receivers is strongly related to the orbital regime of the spacecraft. For LEO satellites, the use of GPS can provide orbital determination solutions with errors in the order of few meters. For MEO, high Earth orbit (HEO), and GEO satellites, the accuracy is reduced up to tens of meters, and for GNSS-based orbit determination around the Moon, accuracy is in the order of hundreds of meters or even few km, depending on the geometry of the mission and the characteristics of the spacecraft. A summary of the expected accuracy is reported in [Table 6.1](#).

Multiconstellation GNSS receivers

Multiconstellation GNSS receivers can get and process signals coming from different GNSS constellations (i.e., GPS, GALILEO, GLONASS, etc.) to determine the spacecraft state. A multiconstellation GNSS receiver has the

Table 6.1 Typical GNSS accuracy.

Scenario	GNSS accuracy—Order of magnitude
LEO	10^0 m
MEO–HEO–GEO	10^2 m
Moon	$10^3 - 10^4$ m

advantage of having a greater number of satellites in view with respect to a single constellation receiver. With the extra information provided by these new satellites, the solution obtained is not only more accurate but also more robust, since it is able to provide information even if the signal from a certain satellite is blocked by the environment (e.g., parts of the spacecraft blocking the antennas line of sight, Earth, etc.).

As opposed to older single constellation GNSS receivers (e.g., GPS receivers), in which four satellites in view are required to provide a solution, the multiconstellation receivers require five satellites to be in view. Namely, four of them are required to solve the position-time problem, and the extra satellite is needed to calculate the time difference between two constellations.

Table 6.2 summarizes the strengths and weaknesses of GNSS sensors for spacecraft orbit determination. As already discussed, GNSS offers high-accuracy measurements and autonomy capabilities. Furthermore, the development for Earth-base applications offers several solutions and a consolidated hardware design. On the contrary, the position of the GNSS constellation satellites poses an intrinsic limit to the applicability of such technology to orbits higher than GEO band.

GNSS sensor model

GNSS sensor models are based on the Earth-centered inertial (ECI) to Earth-centered Earth-fixed (ECEF) reference frame conversion. The true

Table 6.2 GNSS strengths and weaknesses.

Strengths	Weaknesses
High accuracy	Limited applicability regions (i.e., GNSS constellations shall be in view)
Extensive development	
Consolidated hardware design	
Possibility of autonomous on-board use	

inertial position and velocity, generated in an orbital propagator, are first affected by a digital white noise and a drift brown noise, representative for the internal GNSS signal electronic processing. Then, they are converted in the Earth-fixed reference, which is the GNSS reference frame. A clock model is also present, accounting for internal clock errors. All these quantities are delayed by a variable time latency and sent to output. The velocity measurement is typically further delayed with respect to the position one; this effect is included in the model as a dedicated velocity latency. An example GNSS sensor model implementation in MATLAB/Simulink is shown in Fig. 6.13.

Ground-based orbit determination

Ground-based orbit determination techniques are based on measurements performed by ground stations on the spacecraft. This method usually relies on the use of GSN, a set of ground-based antennas transmitting radio frequency signals to the spacecraft. The most common measurements derived from the processing of these signals are:

- *Range*: a measure of the range between a GSN station and the user spacecraft. Similarly, to GNSS measurements, the range is obtained considering the time it takes for a radio signal to travel from the GSN station to the spacecraft and back to the station.
- *Doppler*: it represents a direct measurement of line-of-sight velocity of a spacecraft relative to a tracking antenna. Please note that doppler measurements don't provide any information on position or velocity normal to the line-of-sight [16]. Furthermore, the radio signals can be processed in different ways:

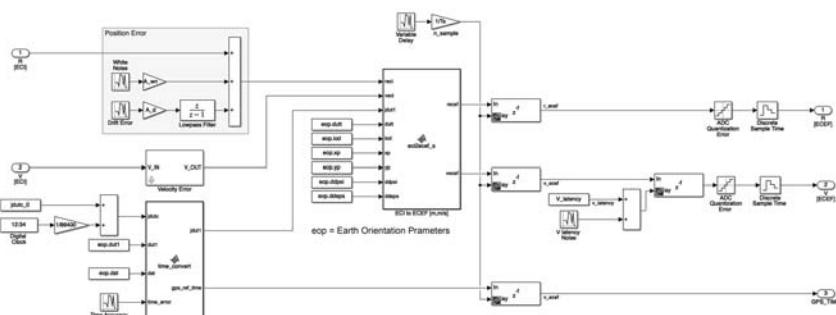


Figure 6.13 GNSS sensor model.

- *One-way*: the spacecraft generates a downlink signal that is transmitted to the GSN. The frequency of the signal is then compared, on ground, against a locally generated frequency.
- *Two-way*: the GSN transmits a signal to the spacecraft. Then, the spacecraft generates and downlinks a signal with a phase coherent with the received signal. Finally, the GSN compares the received frequency with the same reference frequency from which the uplink was generated.
- *Three-way*: the spacecraft is tracked by two stations—one using a two-way mode while the other using a one-way mode.

Please, keep in mind that two-way mode is usually employed for spacecraft navigation.

The basic radiometric measurements provided by the GSN can be also augmented by exploiting the following measurements:

- *Very Long Baseline Interferometry (VLBI)*— Δ -DOR: even if it is not considered as a basic measurement obtained by processing a radio signal, GSN has provided VLBI data since 1980 [17]. The concept of VLBI (or Δ -DOR) is to measure angular separation between a spacecraft and an extragalactic radio source (quasars or galactic nuclei). This is done by processing the radio waves coming from an extragalactic object (ideally close to the spacecraft) and from the spacecraft, in two tracking stations that have a large physical separation. By differentiating the delays of the two incoming signals, the angular separation between the extragalactic source and the spacecraft, in the direction perpendicular to the baseline, can be obtained.
- *Optical data*: usually obtained from a scientific camera available on-board, optical data provide a measure of the direction of a vector from the spacecraft to a target body (e.g., planet, small body). The measured direction can be combined with other range measurements to obtain a complete 3D position of the spacecraft. Angular accuracy of 10 μ rad is expected.
- *Altimetry*: an altimeter provides a measure of the distance between the spacecraft and a target body (e.g., planet, small body). When combined with optical data, altimetry measurements can allow for a complete 3D positioning of the spacecraft. Its operational envelope, however, is limited to distances very close to target body, and, therefore, it is only marginally useful for ground-based spacecraft orbit determination.

In the case of on-ground orbit sensors, the needed infrastructure is usually divided into ground and space segments. This is different with respect to on-board sensors, as the GNSS receivers, which allow obtaining an orbit determination solution without involving ground in the loop.

Ground segment

The ground segment englobes all the ground-based elements of the infrastructure required to perform orbit determination, such as ground stations, mission control, ground network connecting the elements or terminals.

GSNs, such as Deep Space Network from NASA, or ESTRACK from ESA, provide links between the satellites in orbit and the teams on ground. They are in charge of sending and receiving signals with command, information of spacecraft status as well as signals dedicated to navigation purposes [18]. Moreover, the ground segment also includes all the navigation functions to perform orbit determination and process the data to obtain spacecraft position and velocity estimates.

Space segment

Apart from the ground stations on Earth, extra elements are required, in this case on the spacecraft itself, to perform the orbit determination process. Typically, this element is the transponder for two-way range observables. The transponder on-board receives the signals generated by the ground stations, demodulates it, and transmitted it back to ground, using phase modulation on the downlink carrier, making it a signal coherent with the uplink but shifted in frequency.

Ground-based orbit determination accuracy

The typical accuracy of the presented ground-based measurements is reported in [Table 6.3](#).

VLBI clearly improves the overall orbit determination accuracy based only on range and doppler measurements. Optical data are not equally accurate, but provide an additional measurement to retrieve a 3D position. Finally, the use of altimeter improves significantly the position accuracy, but it is limited to close-proximity operations.

Finally, [Table 6.4](#) summarizes strengths and weaknesses of the sensors for ground-based spacecraft orbit determination. As discussed before, ground-based sensors offer high-accuracy measurements in a wide range of possible applicative scenario. However, these solutions imply limited autonomy and communication delays along with a significant operational cost.

Table 6.3 Typical ground-based orbit determination accuracy.

Measurement	Measurement accuracy—Order of magnitude
Range	10^0 m
Doppler	10^{-3} m/s
VLBI	10^1 nrad
Optical data	10^4 nrad
Altimetry	10^{-2} m—dependent on the relative distance

Table 6.4 Ground-based OD strengths and weaknesses.

Strengths	Weaknesses
High accuracy	Limited autonomy
Extensive applicability	High cost
	Communication delays

Attitude sensors

Attitude sensors are used to measure reference directions in the spacecraft body frame, which are then compared with the on-board stored definition of the same directions in the inertial frame. In this way, it is possible to determine the attitude of the spacecraft body frame with respect to the inertial one, as will be discussed in the attitude determination section of [Chapter 9 — Navigation](#).

Many directions could be taken as references, even if the most common attitude sensors measure well-known environmental quantities: star positions, Earth center or limb, Sun direction, or Earth's magnetic field. In fact, there exist four types of sensors that are typically used in attitude determination and control. They are listed below in an approximated order of accuracy:

- Magnetometers.
- Sun sensors.
- Horizon sensors.
- Star sensors.

The following sections describe the different technologies available for each type of sensors, together with the main characteristics, advantages, and drawbacks.

Magnetometers

Magnetometers are a very common type of attitude sensors, flying on most of the satellites in LEO, while they are rarely used in MEO and practically nonpresent in GEO. Similarly, they are not used for interplanetary orbits. This is due to the decreasing of the Earth's internal magnetic field with the square of distance from its center. Thus, for spacecraft orbiting at larger distance from the Earth, the magnetic field is less convenient for both attitude estimation, due to the increased relative error of the sensor, and control, due to the lower ratio of torque over actuator mass. Moreover, magnetometers are also rarely employed in missions around other celestial bodies with a strong magnetic field, since the knowledge of these environments is too poor to have a reliable exploitation of magnetic measurements and torques.

Magnetometers do not provide a direct attitude information or estimation, but they only measure the surrounding magnetic field in magnitude and direction. Since this is a single vector, the satellite state is not fully observable, and it is necessary to combine this measurement with other sources of information to derive the complete attitude. This can be done in several ways, usually using a subset of the below options:

- Other attitude sensor measurements, as Earth sensors or Sun sensors.
- A magnetic field model provided the satellite position is known, either by orbit propagation or estimation using a GNSS [19,20].
- Gyro measurements, together with attitude propagation and other observation techniques.

Beside for estimation of the attitude, the magnetometers are used as well to properly drive magnetic actuators (i.e., magnetorquers in [Chapter 7](#) – Actuators), since the magnetic torque is the function of both the satellite magnetic dipole and the local magnetic field. Hence, a spacecraft with magnetic actuation shall have on-board measurements of the surrounding magnetic field. Note that an on-board stored magnetic field model is not needed to compute magnetic torque commands.

Another use of magnetometers is to provide a coarse angular rate estimation by time derivation of the computed magnetic field unit vector [21]. This operation is usually performed in combination with other sensors, again due to the observability problem of the satellite state.

Magnetometers for attitude measurements are those of vector type, which are capable to measure the vector components of a magnetic field. In fact, they are typically composed by at least three orthogonal single-

axis magnetometers, as in Fig. 6.14. The most common typologies of vector magnetometers for space applications are:

- Magnetoresistive sensors, made of a nickel–iron alloy with a high magnetic permeability, whose electrical resistance varies with a change in magnetic field. They have a very quick response time, but their accuracy is moderate, ranging from 1° to few degrees in direction, and 10 to 50 nT of resolution.
- Fluxgate magnetometers, made of magnetically susceptible cores wrapped by two coils of wire as shown in Fig. 6.15. One coil is connected to an alternating electric current, which drives the cores through opposite alternating cycles of magnetic saturation. If no external magnetic field is present, the fields in the two cores cancel each other and the induced field in the secondary sensing coil would be zero. However, when the cores are exposed to a background field, the one in alignment with the external field has an increased magnetic induction. The contrary happens in the core opposite to the background field. In this way, an alternating electric signal could be induced in the second coil to be measured by a detector. Specifically, the voltage at the secondary coil is the derivative of the flux of the magnetic field vector. In order to achieve meaningful output, the two cores shall reach magnetic saturation during the

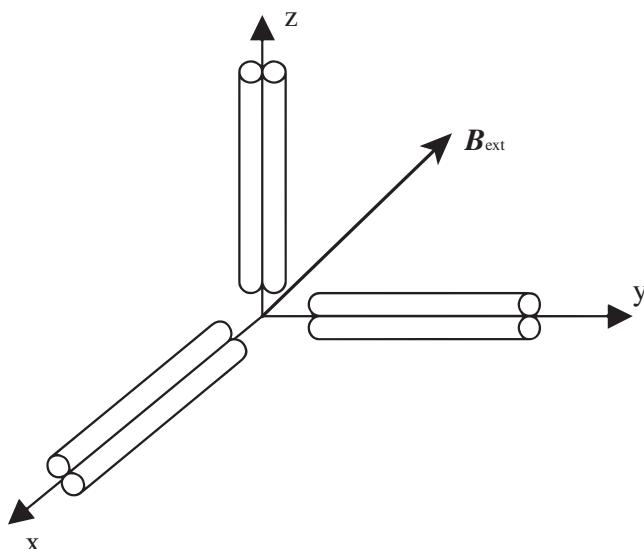


Figure 6.14 Three-axis magnetometer.

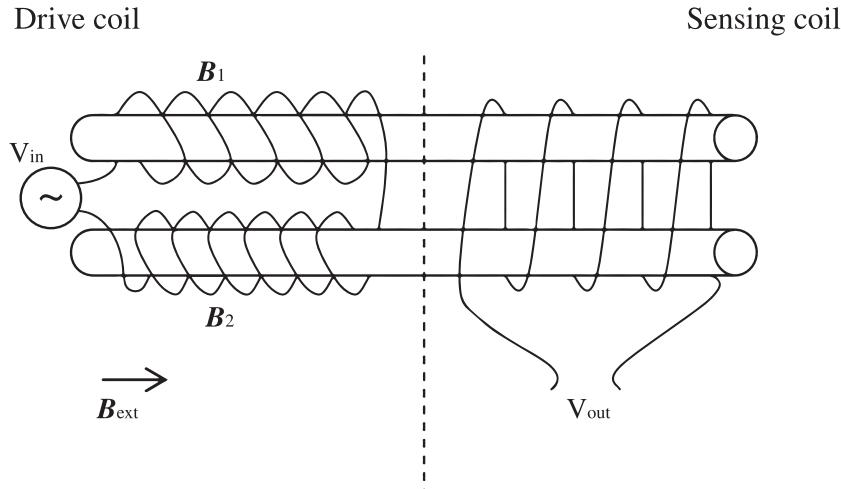


Figure 6.15 Fluxgate magnetometer.

alternating cycle. Hence, the driving current in the primary coil shall consider the expected value of the external magnetic field. Fluxgate magnetometers are the most common for spacecraft applications, since they are relatively small, lightweight, with no moving parts, and inexpensive. They have a quick response time and a good accuracy in the order of 0.1° in direction, with a resolution of 0.1 to 10 nT.

- Microelectromechanical systems (MEMSs) magnetometers for very small-scale applications. This typology of sensor is usually operated by detecting the effects of the Lorentz force or the quantum mechanics effects of the electron tunneling. These sensors are extremely light and power efficient, but they are less accurate with respect to fluxgate magnetometers, with an accuracy in the order of 1° .

Although the accuracy of the measured magnetic field direction is not incredibly high (e.g., ranging approximately from 0.1° to several degrees, depending on the technology used), they are very reliable sensors. Thus, they are typically used in magnetic angular rate damping safe modes (cfr. [Chapter 14 – Applicative GNC Cases and Examples](#)). Another reason for their popularity among GNC systems is that they are quite affordable in terms of costs and on-board resources, compared to other satellite equipment.

The magnetic field measured by the sensor is the composition of Earth magnetic field and the magnetic field generated by the satellite itself. For

this reason, magnetometers accommodation must be properly studied in order to guarantee a level of magnetic cleanliness better or comparable to the sensor performance. Magnetometers can be accommodated both internally and externally to the satellite body, possibly on a boom to increase the distance between the sensor and the satellite body. This is to take advantage of the $1/r^3$ falloff of a magnetic dipole field.

The internal magnetometers accommodation shall be done in a part of the satellite distant from magnetic sources (e.g., attitude control torquers, dipoles, ferromagnetic materials, power lines, current loops in solar arrays, electric motors, etc.). As a rule of thumb, a local magnetic field has to be avoided comparable to the magnetometer bias, in order not to reduce the sensor accuracy. As a mitigation action, it is possible to calibrate the sensor to compensate for this error. However, it is difficult to estimate the actual disturbance while on orbit, since it can be different from what is measured on ground.

If the magnetometer is mounted externally, the sensor is exposed to a more severe space environment. In this case, the main criticalities are the wider range of temperature to be coped with and the increased radiation flux. The shielding to achieve a reduced level of radiation is typically not a problem for analog space qualified sensors, while it may be more problematic for digital sensors or, in general, for commercial components.

Special care is needed to guarantee the compatibility of magnetometers measurement to the use of magnetorquers, which produce a strong and variable local magnetic field, as described in [Chapter 7 – Actuators](#). Indeed, when magnetorquers are activated, they generate a magnetic dipole that can easily saturate the magnetometer (i.e., the generated magnetic field is higher than the magnetometer full scale). This can be solved by scheduling the use of magnetometers and magnetorquers at different times, such that the magnetorquers are commanded with a certain duty cycle and the magnetometers measurements are not acquired at the same time. Again, as a rule of thumb, the time distance between the magnetorquers actuation and the magnetic measurements should be at least three times the magnetorquers characteristic time to allow enough magnetic dipole attenuation.

If the saturation of the measurements is not a problem, the known local magnetic field generated by the magnetorquers can be compensated in the magnetometer's output.

[Table 6.5](#) summarizes the strengths and weaknesses of magnetometers.

Table 6.5 Magnetometers strengths and weaknesses.

Strengths	Weaknesses
Reliable	Poor accuracy ($0.5\text{--}5^\circ$ approximately)
Affordable	Measurement influenced by satellite magnetic cleanliness
Measurement always available on the orbit	Not suited for orbits above LEO
Used both for attitude estimation and magnetic torquers driving	No direct attitude information

Sun sensors

A Sun sensor is a device that senses the direction of the Sun with respect to the sensor reference frame.

The Sun sensor is operated based on the entry of light into a thin slit or a pinhole on top of a chamber whose bottom part is lined with a group of light-sensitive cells. The chamber casts an image of the aperture on the detector. The cells at the bottom measure the distance of the image from a centerline and determine the refraction angle by using the chamber height.

The receptive cells convert the incoming photons into electrons and, thus, voltages, which are in turn converted into a digital signal. When two groups of light-sensitive cells are placed perpendicular to each other, the direction of the Sun with reference to the sensor axes, and consequently the spacecraft body frame, can be calculated.

The design of a Sun sensor system for any spacecraft application must provide acceptable physical characteristics, meet performance specifications, and cope with interference sources and environmental conditions while minimizing penalties to spacecraft design and performance. Physical characteristics include size, weight, and power requirements. The major performance parameters are accuracy, stability of the measurement, field of view, transfer function characteristics, and resolution. Interference sources include sunlight reflected from Earth and other planetary bodies, reflection from spacecraft surfaces, and spacecraft electromagnetic fields [22]. In particular, Earth albedo determines a significant disturbance in the Sun sensor measurements, and it shall be calibrated in the GNC algorithms to achieve the full sensor performance. Hence, GNC functions shall be capable to estimate the incoming light radiation from the Earth to be subtracted from the sensor measurements.

The purpose of this chapter is to highlight the most important characteristic that a GNC engineer needs to consider when designing a system entailing Sun sensors. Here, we limit the description to sensors that indicate the orientation of the Sun with respect to a known reference frame. The orientation is sensed by detecting the intensity difference between radiation arriving from the solid angle subtended by the Sun boundaries and the rest of the radiation reaching the sensor from the adjacent regions surrounding the sensor's field of view.

Sun sensors are usually classified into three types:

1. Analog sensor whose output signal is a continuous function of the angle of incidence.
 - a. Coarse Sun sensors (CSSs).
 - b. Fine Sun sensors (FSSs).
2. Digital sensor that produces encoded discrete output of the function of the angle of incidence.
3. Sun presence sensor that provides a constant output signal when the sun is in field of view.

Analog sun sensors

The analog Sun sensors deliver a continuous function based on the incident radiation. The analog sensors are furtherly classified in CSSs and FSSs.

Coarse sun sensors

These types of analog sensors measure the amount of incident light without using windows nor projections. Incident light flux is proportional to the cosine of the angle between the sun and the normal vector of the light-sensitive cell, which allows us to calculate the angle of incidence of the solar vector. With reference to Fig. 6.16, the equation simply reads:

$$J = J_s \cos(\alpha)$$

where J is the solar intensity at the sensor, J_s is the Sun flux light constant, and α is the incidence angle between the light direction and the normal to the sensor.

Solar sensors of this type only use one photodiode, and the electric current they create is used to deduce the angle of incidence. At this point, it is insightful to think of the solar panels as a particular type of CSSs because of their capability to generate current following the cosine law mentioned before.

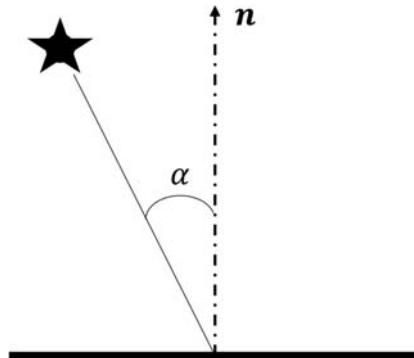


Figure 6.16 Lambert cosine law.

If we assume the Sun as a light point source identified by the \mathbf{s}_b unit vector from the spacecraft to the Sun, the output of a CSS mounted on the face A , whose normal is identified by \mathbf{n}_A , is [23]:

$$I_A = \begin{cases} I_{\max}(\mathbf{n}_A \cdot \mathbf{s}_b) & \text{for } \mathbf{n}_A \cdot \mathbf{s}_b > 0 \\ 0 & \text{for } \mathbf{n}_A \cdot \mathbf{s}_b < 0 \end{cases}$$

where I is the CSS output current. Let us now assume that we have another sensor on the other side of the spacecraft, namely on face $-A$, whose normal is identified by $-\mathbf{n}_A$, then:

$$I_{-A} = \begin{cases} I_{\max,-A}(\mathbf{n}_{-A} \cdot \mathbf{s}_b) = -I_{\max}(\mathbf{n}_A \cdot \mathbf{s}_b) & \text{for } -\mathbf{n}_A \cdot \mathbf{s}_b > 0 \\ 0 & \text{for } -\mathbf{n}_A \cdot \mathbf{s}_b < 0 \end{cases}$$

Hence, we can write:

$$I_A - I_{-A} = I_{\max}(\mathbf{n}_A \cdot \mathbf{s}_b), \forall \mathbf{n}_A \cdot \mathbf{s}_b$$

Adapting the exhaustive example in Ref. [23], given the above relationships, if we equip the spacecraft with at least 6 not coplanar Sun sensors, $\pm \mathbf{n}_i, \pm \mathbf{n}_j, \pm \mathbf{n}_k$:

$$\begin{bmatrix} I_i - I_{-i} \\ I_j - I_{-j} \\ I_k - I_{-k} \end{bmatrix} = I_{\max} \begin{bmatrix} \mathbf{n}_i \cdot \mathbf{s}_b \\ \mathbf{n}_j \cdot \mathbf{s}_b \\ \mathbf{n}_k \cdot \mathbf{s}_b \end{bmatrix} = I_{\max} \begin{bmatrix} \mathbf{n}_i^T \\ \mathbf{n}_j^T \\ \mathbf{n}_k^T \end{bmatrix} \mathbf{s}_b$$

The Sun unit vector can be computed as:

$$\mathbf{s}_b = \frac{1}{I_{max}} \begin{bmatrix} \mathbf{n}_i^T \\ \mathbf{n}_j^T \\ \mathbf{n}_k^T \end{bmatrix}^{-1} \begin{bmatrix} I_i - I_{-i} \\ I_j - I_{-j} \\ I_k - I_{-k} \end{bmatrix}$$

These sensors are very sensitive to variations in temperature, which can cause differences in the current generated, and the estimated angle is directly proportional to this current, leading to imprecise readings of the angle of the solar vector. They are not very precise when the incident angle is close to perpendicular. For this and other reasons, coarse sensors are not as accurate as fine sensors.

Fine sun sensors

In Fig. 6.17, which represents a single-axis fine analog Sun sensor, two photosensitive elements are placed close to each other. The Sun rays are collimated or deviated to form an apparent image on the elements. Each element outputs an electrical current of different intensity based on the Sun direction. The difference between the current outputs developed across the two detector elements yields the sensor output. Whenever the two currents are equal, meaning that the Sun illumination on the two elements is equal, the sensor output is zero and the sensor reached the null point, with the Sun perpendicular to the sensor.

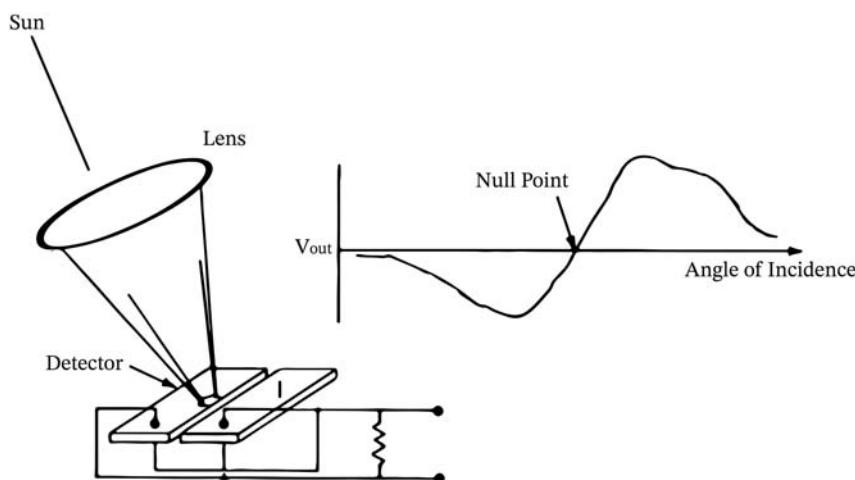


Figure 6.17 Analog Sun sensor.

Digital Sun sensors

In Fig. 6.18, the working principle of a digital Sun sensor is schematized. The Sun is imaged as a line across an array of separate elements. Each element produces a 1 or 0 binary bit in the multichannel output, depending on whether light reaches the element through the mask and whether the sensor's output in each channel exceeds threshold values established in associated circuitry. The binary number assigned to the channel identifies the position of the light imaged on the array of elements. The increased number of photosensitive units makes them more sensitive, increasing the accuracy of the sensor. Modern digital Sun sensors exploit multipixel optical detectors to further increase the sensor accuracy.

As a GNC engineer, an important step is the modeling of the sensors as a mathematical equation. We have seen that the Sun's relative position is sensed with respect to the spacecraft body frame. This unit vector, called Sun measurement vector, is output by the sensor in the body reference frame. To determine the spacecraft attitude, it is necessary to determine the rotation between a known reference frame and the body reference frame $b\mathbf{A}_i$.

Thus, the sensor can be modeled, according to sensor modeling section, as:

$$\mathbf{s}^b(t) = b\mathbf{A}_i \mathbf{s}^i(t) + \boldsymbol{\varepsilon}(t),$$

where \mathbf{s}_b is the Sun vector in the body frame, and \mathbf{s}_i is the Sun vector in the inertial reference frame, which is typically adopted as reference and $\boldsymbol{\varepsilon}$ the expected sensor errors. At this point, it becomes clear that an attitude estimation algorithm that uses the Sun sensors measurements requires the

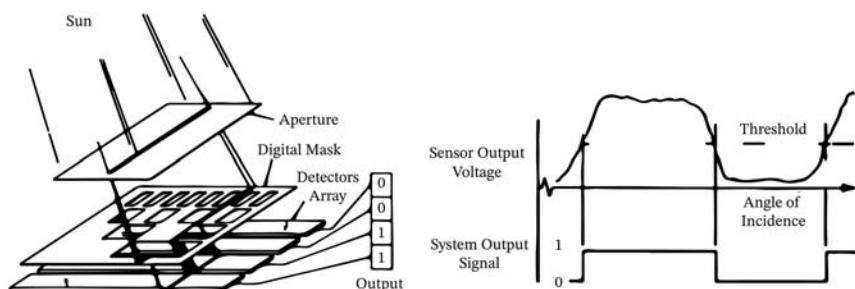


Figure 6.18 Digital Sun sensor.

knowledge of the position of the Sun with respect to a known reference frame, which is typically an inertial one. Such information can be retrieved from Sun ephemerides or from a simplified model of the Sun motion with respect to the spacecraft orbital reference frame [24].

To summarize, the most important qualitative characteristics of the various types of Sun sensors is reported in [Table 6.6](#).

In general, Sun sensors can be quite accurate, i.e., roughly $<0.01^\circ$, but high-resolution FSSs can reach up to 1 arcmin resolution. Unfortunately, especially for LEO orbits, they are subject to occultation during eclipse times. For this reason, when designing a system based on Sun sensors, it is remarkably important to consider the shortage provoked by the loss of this signal without degrading the GNC performance (e.g., using an alternative sensor or propagating the attitude state with a gyroscope). Furthermore, Sun sensors need clear fields of view.

[Table 6.7](#) summarizes the strengths and weaknesses of Sun sensors.

Sun presence sensors

Analog and digital Sun sensors are more complex compared to the Sun presence sensor, which delivers a simple binary output. Indeed, Sun presence sensors are only capable to detect if the Sun is in front of the sensor, within a certain angle, or not. This sensor typology may be helpful to implement fast Sun searching or Sun avoidance algorithms.

Sun sensor model

Sun sensor models are based on a modeling approach rotating the ideal spacecraft-to-Sun unit vector in the sensor reference frame, aligned with

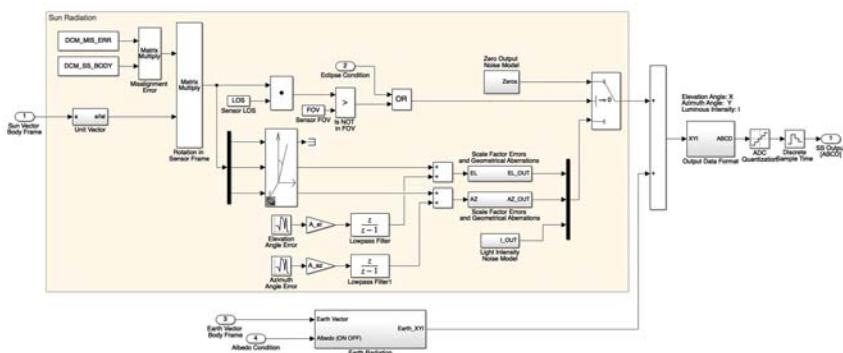
Table 6.6 Types of Sun sensors.

Type	Features
Coarse Sun Sensors	It only provides the relative intensity of the incident light rays according to the cosine law.
Fine Sun sensors	Digital <ul style="list-style-type: none"> • Digital output • No analog detection circuitry • Less noise • More power consumption
	Analog <ul style="list-style-type: none"> • Analog output • Less power consumption

Table 6.7 Sun sensors strengths and weaknesses.

Strengths	Weaknesses
Reliable	Not working during eclipses
Affordable	No direct attitude information
It can be used in a large portion of the Solar System	Affected by perturbing light sources
Good accuracy	

its line of sight. In the case the Sun is within the sensor's field of view, the Sun vector is processed adding observation errors affecting both the Sun angles and the luminous intensity. The angles are affected by brown electronic noise, distortions due to geometrical aberrations, and scale factor errors. Finally, the Sun vector is converted in the output data format, as described in the sensor datasheet, thanks to some geometrical transformations. The possible luminous disturbance introduced by the Earth's albedo is also modeled. An example digital Sun sensor model implementation in MATLAB/Simulink is shown in Fig. 6.19. A similar modeling approach can be also used to model Earth horizon sensors or Star sensors. In the latter case, the Star sensors can estimate the complete attitude of the spacecraft, by comparing the visible stars with their positions saved in a star catalog, as will be discussed in the star sensors section. Thus, the star sensor model shall also include the star catalog, the star matching process, with possible identification errors, and the final attitude reconstruction.

**Figure 6.19** Sun sensor model.

Horizon sensors

Horizon sensors (often referred specifically to Earth sensors) are infrared devices that can detect the discontinuity in temperature between the deep space and the Earth (or planetary) atmosphere. An important task that required considerable effort during the historical development of these types of sensors is the determination of the best spectral region for defining the space-to-Earth discontinuity and providing robustness to disturbance radiation. Most frequently, sensors use the Earth's radiation in the infrared spectrum (from 2 to $30\mu\text{m}$) and, in particular, within the narrow $14\text{--}16\mu\text{m}$ [23,25]. The reasons why horizon sensors for Earth limb detection are often centered in the above range are:

- The variation between maximum and minimum radiance is smaller compared to the visible spectral band.
- When imaging the Earth in the infrared spectrum, the terminator line on the Earth surface vanishes. This is because the difference between night and day temperature is negligible with respect to the absolute temperature scale (i.e., with reference to the absolute zero).
- Imaging in the infrared spectral band prevents the loss of signal during eclipses.

Earth presents a finite size and cannot be interpreted as a point because the approximation would just be too erroneous. Indeed, the solid angle subtended by the Earth is roughly 3.9 steradians at 500 km altitude, compared to the approximation of the Sun, which extends for nearly $7 \cdot 10^{-5}$ steradians [26]. In general, the most important parts of a horizon sensor are the infrared optical system, including the spectral filter, and the focal plane where radiance detectors are placed, e.g., thermistors or thermopiles, as shown in Fig. 6.20.

Horizon sensors are classified into two distinct typologies:

1. Static sensors: they are mounted fixed on the spacecraft in predetermined body axes directions. For this reason, this kind of sensors is dedicated to spacecraft that have always (or for a large part of the time) the Earth within the sensor field of view. A typical end-to-end pipeline of the sensor processing is shown in Fig. 6.21.
2. Scanning sensors: they are composed of a moving detector, whose field-of-view (FOV) scans the Earth. Scanners are sensors that either mechanically, electronically, or passively (on a rotating vehicle) scan a large volume of object space with a scan pattern fixed relative to the sensor or vehicle [26].

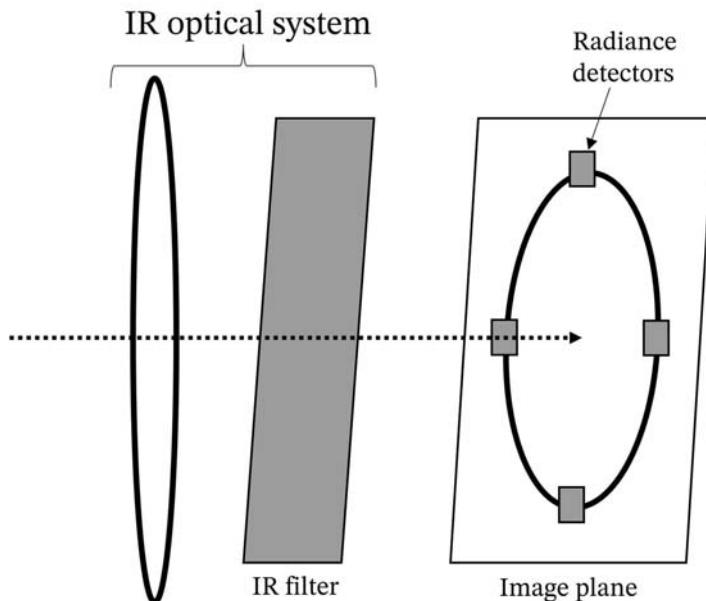


Figure 6.20 Schematic of a horizon sensor.

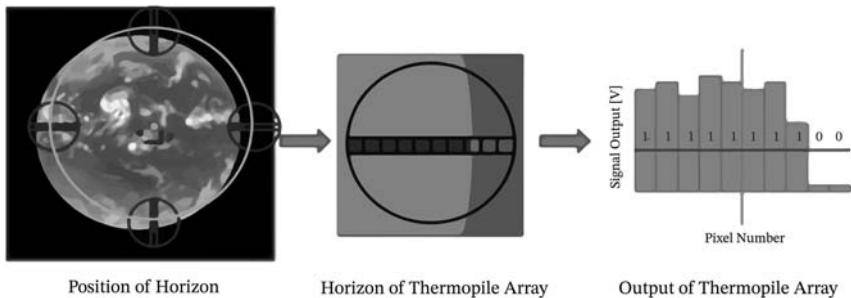


Figure 6.21 Workflow of Earth horizon sensor functioning.

The accuracy of horizon sensors ranges from 0.1 to 0.5° [23,26].

As already mentioned, the Earth sensor delivers a measurement of the relative attitude between the Earth and the spacecraft. Given the shape of the Earth (circular as a first approximation), the output of a horizon sensor inherently possesses an ambiguity in the yaw angle. In other words, the rotation around the radial axis of the comoving local-vertical-local-horizontal (LVLH) frame (cfr. Chapter 2 – Reference Frames and Planetary Models)

is not determined. Only, roll and pitch angles can be estimated by the horizon sensor. In its simplest form, the model of an Earth horizon sensor reads:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\phi}_{b,E} \\ \dot{\theta}_{b,E} \end{bmatrix} = \begin{bmatrix} \dot{\phi}_{b,LVLH} \\ \dot{\theta}_{b,LVLH} \end{bmatrix} + \begin{bmatrix} \alpha_{LVLH,E} \\ \beta_{LVLH,E} \end{bmatrix} + \boldsymbol{\epsilon},$$

where $\phi_{b,LVLH}$ and $\theta_{b,LVLH}$ are the true roll and pitch attitude of the satellite, i.e., the spacecraft body frame with respect to the LVLH frame; $\alpha_{LVLH,E}$ and $\beta_{LVLH,E}$ are the angles between the Earth's horizon and the LVLH frame. Finally, $\boldsymbol{\epsilon}$ is the measurement error. For circular orbits and under the assumption of a spherical Earth, the angles between the Earth horizon and the LVLH comoving orbital frame are constant and equal to:

$$\alpha_{LVLH,E} = \beta_{LVLH,E} = \pi - \sin^{-1}\left(\frac{R_E}{R_E + h}\right)$$

where R_E is the Earth radius and h is the orbital altitude.

[Table 6.8](#) summarizes the strengths and weaknesses of horizon sensors.

Star sensors

Star sensors (or star trackers) are devices that image the celestial sphere and determine the attitude of the spacecraft with respect to the inertial reference frame, where the positions of the stars are catalogued. Star sensors are basically optical cameras, see [Fig. 6.22](#), which use detectors to react at the incoming light from the stars. The stars' directions are extrapolated from the image plane to the spacecraft body-fixed reference frame. Each star ephemeris is stored in a preloaded catalog; hence, the correspondence between the imaged stars and database ones delivers an estimation of the spacecraft attitude. Indeed, by measuring more independent directions, star sensors can

Table 6.8 Horizon sensors strengths and weaknesses.

Strengths	Weaknesses
Roll and pitch attitude information	It works well in the vicinity of planets or, in general, for planetary pointing mission (e.g., Earth nadir pointing)
Good accuracy	Complex calibration of spectral region
Not affected by eclipses	Not complete attitude information

completely solve the attitude problem without the need of other primary sensors. This process and the relevant determination algorithms to work out the attitude problem are detailed in [Chapter 9 – Navigation](#).

According to Ref. [27], a star sensor comprises an imaging function, a detecting function, and a data processing function. The imaging function collects photons from objects in the field of view of the sensor and focuses them on a detecting element. This element converts the photons into an electrical signal that is then subject to some processing to produce the sensor output. International space standards [27] thoroughly define the classification and capabilities of different star sensors. Hereby, a brief summary on the most important capabilities of star sensors is reported:

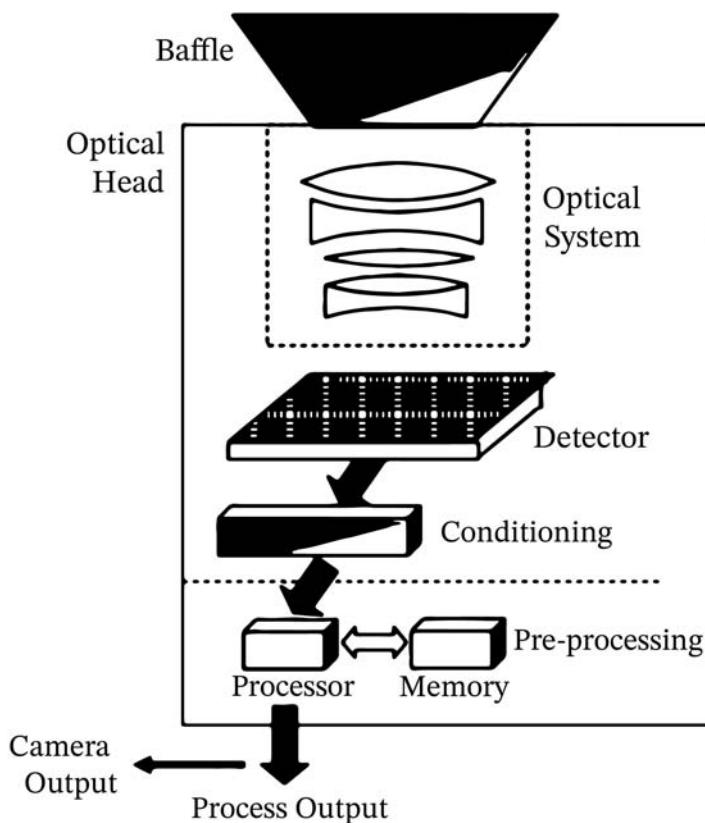


Figure 6.22 Schematic of star sensor elements [27].

- *Cartography*: this function entails the capability of delivering the star position and measurement date for each detection. Star position is identified in the sensor reference frame, whose rotation with respect to the space-craft is known.
- *Autonomous attitude determination*: this function entails the capability of delivering the orientation of the sensor reference frame with respect to the adopted inertial reference frame. According to ECSS [27], the sensors implementing such functionality should deliver a validity flag of the determined attitude. This capability is often referred as the *lost-in-space problem* solution. Initial attitude acquisition computes at least three centroids for the brightest clusters of pixels, representative of stars. A set of descriptors for these detections groups, comprising brightness and arc lengths are generated and used in the pattern searching algorithms to match the detections with known stars.
- *Star tracking*: this function entails the capability of delivering the position of selected stars with respect to a sensor-attached reference frame. This capability does not imply to autonomously identify the stars to be tracked or explicitly identified by the unit. Indeed, the initial selection of the star images to be tracked is not autonomous and must be done offline. However, it maintains the identification of each star image and correctly updates the coordinates of each star image as it moves across the detector due to the angular motion of the sensor. In this way, it allows for faster attitude problem solutions by comparison with respect to previously known attitude states.

Stars are highly accurate references and nearly independent by the orbit position. Furthermore, they are generally available everywhere in the sky. For these reasons, star sensors are among the best attitude sensors in terms of accuracy, which falls below few arcseconds in the sensor boresight direction and slightly larger errors for rotations around the boresight axis. Nevertheless, the increased accuracy comes with considerable cost, in terms of mass and power consumption with respect to Sun and horizon sensors. Star sensors require time in solving the initial attitude problem from a lost-in-space condition, and they are generally slower than other attitude sensors. In fact, during maneuvers, there might be the need of attitude propagation with gyroscopes, since star sensors could lose the attitude solution if large slew rates are present. Moreover, star sensors are delicate since they are disturbed or

even damaged if their field of view is directed to an intense light source (e.g., the Sun or the Earth). For this reason, they are typically equipped with protecting baffles and need to be covered if they are pointed to the luminous object. Finally, their optical elements shall be protected from contaminants, such as thruster plumes.

The geometric modeling of a star sensor resembles the pinhole camera model, which will be thoroughly discussed in [Chapter 9 – Navigation](#). For the sake of clarity, it is here reported the founding relationship of a generic point pinhole projection, as in [Fig. 6.23](#).

Whenever a star is imaged, a block of pixels is illuminated due to the optics being slightly defocused, hence yielding a point spread function distribution rather than a unique pixel. A centroiding algorithm determines the pixel location of the star by calculating a weighted average of the pixels' brightness composing the block. The resulting detection accuracy depends on the sensor, the star brightness, the exposure time, and various errors related to the implementation of the optics [23]. For a GNC engineer, not strictly involved in the manufacturing of these sensors, it is important

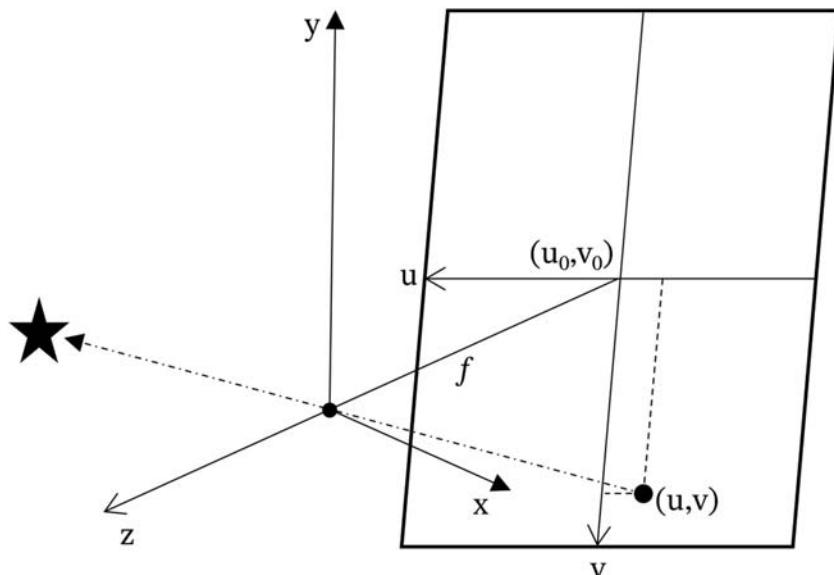


Figure 6.23 Star sensor modeling under the assumption of pinhole camera. The reference frame x, y, z is the sensor frame; u, v is the image frame and f is the focal length.

to obtain all the data (e.g., from the supplier) to characterize the sensor performance. A generic measured star vector \mathbf{s} , namely the unit vector from the spacecraft to the identified star, can be computed with reference to Fig. 6.23 as:

$$\mathbf{s} = \frac{1}{\sqrt{(f^2 + (u - u_0)^2 + (v - v_0)^2)}} \begin{bmatrix} u - u_0 \\ v - v_0 \\ f \end{bmatrix} + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon}$ is the measurement error. The star catalog used in a star sensor is a critical variable that largely affects the accuracy of the sensor. Typically, sensors offer an end-to-end pipeline depending on the capability the sensor provides, as mentioned before (e.g., cartography, star tracking, attitude determination). Nevertheless, it is interesting to report how the number of stars influences the sensor output. To make a proper attitude determination, we may need at least $N = 3/4$ stars to be identified in the image. The more objects are identified, the more robust to random errors our estimation will be. The probability of finding at least N stars in the sensor field of view can be assumed as a Poisson distribution [23]:

$$P(N) = e^{-N_{avg}} \frac{N_{avg}^N}{N!}$$

where N_{avg} is the average number of stars we expect in the FOV. The average number of stars in FOV can be determined by the FOV angle and the number of catalog stars $N_{catalog}$, knowing that the size of the celestial sphere is 4π sr:

$$N_{avg} = \frac{N_{catalog} \cdot FOV^2}{4\pi}$$

This means that the probability of finding at least N stars in the FOV is influenced by the number of stars in the catalog and the instrument FOV. Given a certain value of average number of stars in FOV, the probability of finding and detecting at least N stars in the FOV is represented in Fig. 6.24.

For instance, with reference to Fig. 6.24, the probability of finding at least $N = 5$ stars with an average of $N_{avg} = 8$ stars in FOV is 90%. The number of catalog stars or the required FOV can be determined as:

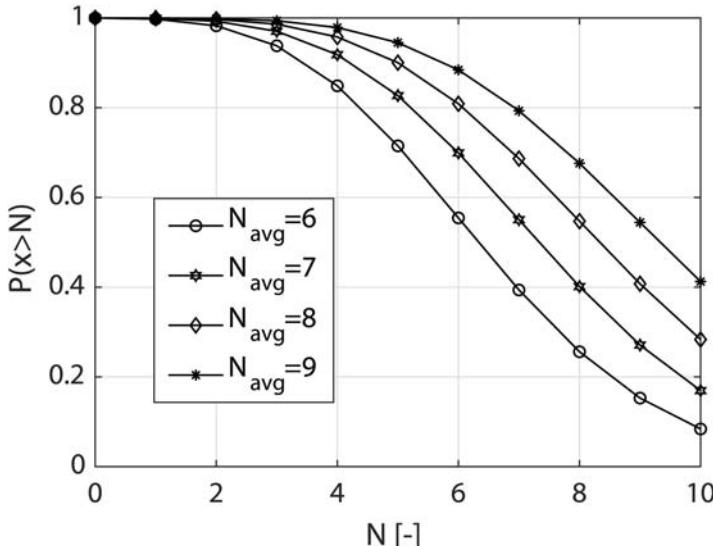


Figure 6.24 Number of detected stars in FOV for different average number of stars in FOV, linearly dependent from the number of catalog stars.

$$FOV = \sqrt{\frac{4\pi N_{avg}}{N_{catalog}}}$$

$$N_{catalog} = \frac{4\pi N_{avg}}{FOV^2}$$

To solve this trade-off, we could introduce a constraint on the resolution of the sensor, which limits the FOV for a given detector array, namely the number of pixels utilized for the image representation. Let us go through an example: roughly speaking, if we have a star sensor with FOV angle equal to 20° and a detector of 1024 pixels, we obtain a resolution of $\epsilon = \frac{FOV}{n_{px}} \approx 0.02 \text{ } px$. We may obtain subpixels resolution depending on the centroiding algorithm that can achieve accuracy of $\gamma = 0.1 \text{ } px$ width. In this case, $\epsilon_{centroid} = \gamma\epsilon$. The final accuracy of the pointing direction depends on the number of stars in the FOV [28], namely $N = 5$ in this example, as:

$$\text{accuracy} = \frac{\epsilon}{\sqrt{(N)}} = \frac{FOV}{n_{px}\sqrt{(N)}} \approx 30 \text{ arcsec}$$

Table 6.9 Star sensors strengths and weaknesses.

Strengths	Weaknesses
Extremely accurate	Mass and volume
Provide complete attitude information	Complexity and high cost
Wide applicability across the Solar System	Must avoid luminous objects (e.g., the Sun)
	Requires time to acquire the initial attitude state from a lost-in-space condition

Or, with centroiding accuracy $\gamma = 0.1 \text{ px}$:

$$\text{accuracy} = \frac{\varepsilon_{\text{centroid}}}{\sqrt{(N)}} \approx 3 \text{ arcsec}$$

[Table 6.9](#) summarizes the strengths and weaknesses of star sensors.

Performance comparison

[Table 6.10](#) summarizes the typical orders of magnitude of the performance for each presented attitude sensor. Obviously, this is only a rough guideline. A GNC engineer should go through the specifications and the datasheets of each component to acquire precise indications on the actual performance data.

Inertial sensors

Inertial sensors are used to transduce inertial forces into electrical signals, to measure the specific forces — i.e., accelerations — and the angular rates experienced by them and, consequently, by the bodies on which they are installed. Inertial sensors sensing linear acceleration are referred to as accelerometers, while the angular motion is sensed by sensors denoted as gyroscopes. A single accelerometer, or gyroscope, is capable to measure the associated quantity along one axis. Thus, a set of three inertial sensors along orthogonal directions is needed to measure the whole 3D quantity, and it is commonly defined as Inertial Reference Unit (IMU). When triaxial accelerations and angular velocities are provided by a single component, this is referred to as IMU, which is a six-axis sensor containing three orthogonal

Table 6.10 Typical order of magnitude of the performance for each presented attitude sensor.

Sensor	Range of performance
Magnetometer	0.5–5°
Sun sensors	0.001–1°
Horizon sensors	0.1°–0.5°
Star sensors	1 arcsec–1 arcmin

accelerometers and three orthogonal gyroscopes. Sometimes, IMUs can be also equipped with three orthogonal magnetometers (cfr. [Section Magnetometers](#)), and, in this case, they are denoted as nine-axis IMU.

Two basic typologies of inertial sensors exist: the “gimballed” and the “strapdown.” The former was the first original application of inertial sensors technology, which uses stable platform techniques, where the inertial sensors are mounted on a stable platform and are mechanically isolated from the motion of the vehicle. In this case, triads of accelerometers and gyroscopes are suspended by a system of three gimbals keeping the platform in a fixed orientation with respect to the inertial space. This can be achieved by exploiting conservation laws or with a gyro-stabilized mechanized system. In mechanized gimballed system, the stabilization is guaranteed by three integrating gyros, whose output is proportional to the rotation angles to which the sensors are subjected. Their measurements are used in a feedback cycle commanding null-seeking servomotors, which drive each gimbal to maintain a constant orientation in inertial space. Gimballed inertial sensors directly measure inertial accelerations, and, thus, the displacement from the initial position is computed integrating twice with respect to time. Analogously, the body rotations are immediately retrieved from the gimbal’s angles with respect to the platform supports. Despite the very high accuracy that is possible to achieve with this typology of inertial sensors, their technological complexity ties down their usage to systems requiring extremely precise measurements for prolonged time, such as ships and submarines. Purely gimbal sensors are practically unrealistic due to the requirement to have almost null friction in the bearings, while mechanized ones have a considerable impact on mass, volume, power, costs, and maintenance of the system. Moreover, gimballed systems are affected by the gimbal lock, which happens

when the axis of two of the gimbals are driven in the same direction, losing one degree of freedom on the 3D isolation capability (also experienced by the Apollo 11 Moon mission [29]). This problem may be overcome by using a fourth gimbal or by performing dedicated reset rotations. Both solutions further increase the complexity of gimballed inertial sensors.

For all these reasons, from the early 1970s [30,31], the inertial sensors technology was pushed toward alternative, simpler, gimbaless inertial sensors. The basic idea is to rigidly attach, or “strap down”, gyroscopes and accelerometers directly to the system body, using the gyroscope information to compute the current attitude of the body and, then, generate a coordinate transformation that operates on the accelerometer data to give acceleration resolved into inertial axes. In effect, this results in a “mathematical gimbal set,” replacing the mechanical one. The theoretical aspects behind strapdown inertial sensors are two: the measurement of acceleration in body-fixed coordinates and the determination of the orientation of the body-fixed reference frame with respect to inertial one from gyro information. Strapdown accelerometers measure real inertial acceleration, but they resolve it in body-fixed axes. Thus, apparent accelerations arise interpreting these components in the noninertial reference frame. In general, strapdown systems demand for increased computing capabilities, since the attitude transformation computations have to be carried out at a very high frequency. The lack of the inertial platform let the body-mounted instruments experience both the angular and linear motions of the system and most of its vibrations. Moreover, these instruments are subject to the entire maneuver envelope of the spacecraft, and not only to the residual drift rotations of the gyro-stabilized platform. For these reasons, they shall be capable to detect much higher rates of turn, while maintaining good drift accuracy values. Consequently, very broad and linear dynamics range is mandatory for this kind of inertial sensors. Despite these minor drawbacks, strapdown inertial sensors are now widely used in spacecraft applications, and with the current on-board computing capacities, they are a fully mature technology, with performance almost comparable to the gimballed systems.

Gyroscopes measure the changes in vehicle attitude or its angular rate with respect to an inertial reference frame. Accelerometers, however, are not capable to distinguish among the total acceleration of the vehicle with respect to inertial space, and that caused by the presence of a gravitational field. The output of these sensors is the difference between the true

acceleration in space and the acceleration due to gravity. Namely, the nongravitational force per unit mass exerted on the instrument:

$$\mathbf{a}^b = {}_b\mathbf{A}_i \left(\ddot{\mathbf{r}} - \mathbf{g}(\mathbf{r}) \right) \quad (6.2)$$

where the superscript b indicates that it is measured in the body reference frame, \mathbf{r} is the inertial position vector of the accelerometer, $\mathbf{g}(\mathbf{r})$ is the position-dependent gravitational acceleration, and ${}_b\mathbf{A}_i$ is the direction cosine matrix rotating a vector from the inertial to the body reference frame. Hence, in free-fall orbiting applications, accelerating due to the gravity of celestial bodies only, an ideal accelerometer placed at the center of gravity of the spacecraft would measure zero. Then, to compute the total acceleration acting on the body, which is needed to propagate its motion, a gravitational field model shall be used to correct accelerometer measurements with gravity accelerations. The navigation functions, discussed in [Chapter 9 – Navigation](#), exploiting strapdown inertial sensors measurements shall combine body rotation estimation, reference frame transformation, gravitational acceleration evaluation, translational and rotational coupling correction, and dynamics integration, as in [Fig. 6.25](#).

Accelerometer's technology is well established since many decades, and it is based on mechanical sensors or solid-state ones. Despite the construction principle of these sensors, they are usually based on Newton's second law of motion to measure the force acting on a small mass, known as a proof or seismic mass, which is contained within the vehicle, and it is typically connected via a spring or an oscillating arm to the case of the instrument. When

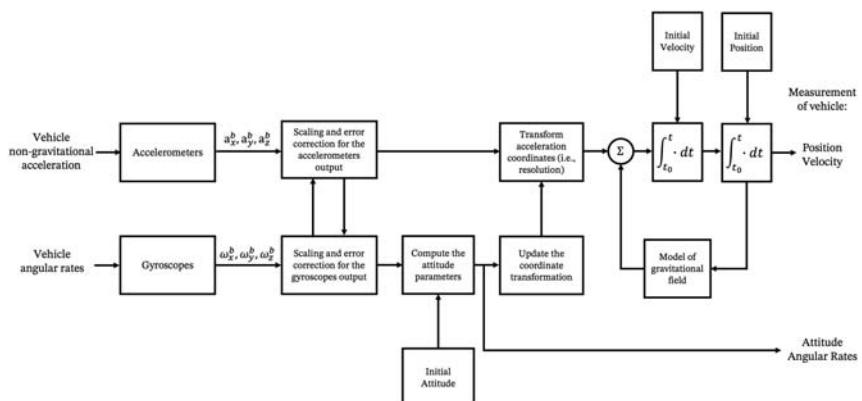


Figure 6.25 Functional components of a navigation system based on inertial sensors.

the instrument is subjected to an acceleration along its sensitive axis, the proof mass tends to respond to the change in movement according to its own inertia. A transducer is designed to sense the relative movements between the proof mass and the sensor case, converting them in calibrated electric signals proportional to the nongravitational forces acting on the body. As already said, an accelerometer freely orbiting around a celestial body would output a zero measurement, since both the case and the proof mass experience a common influence of the surrounding gravitational field. In this case, the acceleration of the instrument with respect to an inertial reference frame is $\mathbf{a} = \ddot{\mathbf{r}} = \mathbf{g}$, and the specific force is zero in accordance with Eq. (6.2). The principles of operation of accelerometers depend on the construction technology, which can be based on: force-feedback pendulum, optical fiber pendulum, vibratory quartz crystal technology [32], acoustic wave resonator on piezoelectric cantilever beam [33], silicon-based pendulum [34], interferometry [35], optical waveguides and Bragg gratings [36]. In recent years, MEMS technology began to be popular for inertial sensors. This construction principle will be discussed in the following, together with gyroscope MEMS technology, but high-accuracy accelerometers will continue to incorporate mechanical sensors, with some use of resonant devices [32].

Gyroscope's technology is typically divided in two broad classes: rate gyros that read angular rates and rate-integrating gyros that measure integrated rates or angular displacements. The most basic gyroscope construction principle is based on the angular momentum conservation principle of a wheel, or rotor, spinning at high speed. In this case, the instrument contains a spinning mass, whose angular momentum vector, coincident with the axis of spin of the rotor, remains fixed in the inertial reference frame. When the sensor is subjected to a rotation along its sensitive axis, a precession displacement between the instrument case and the spinning axis is sensed by a transducer converting it in a calibrated electric signal. In alternative, the precession principle can be also applied to accurately measure the rotation rates, by applying a torque to maintain the angular momentum of the proof mass fixed in the body frame. Then, the measurement of this torque provides measurements of the angular velocity of the instrument, according to:

$$\mathbf{t}^b = {}_b\omega_i^b \times \mathbf{h}^b, \quad (6.3)$$

where \mathbf{h}^b is the gyro's angular momentum, ${}_b\omega_i^b$ is the angular rate of the body with respect to the inertial reference frame, and \mathbf{t}^b is the output torque

measured in body frame. Single-axis gyros sense only one direction of the torque, and the other reaction torque is provided by mechanical constraints on the orthogonal axis. Very accurate spinning mass gyros are single-axis floated devices, where the rotor is contained in a case that is immersed in a buoyancy fluid to reduce the load on the gimbal bearings [32]. Two-axis gyros are called dry tuned-rotor gyros (DTGs) because they are not floated. They have two input axes which are mutually orthogonal, and which lie in a plane which is perpendicular to the spin axis of the gyroscope. The rotor deflection is sensed on both axes, and it is controlled electromagnetically according to Eq. (6.3). DTGs have come very close to the performance of single-axis floated gyros, but never achieved performance equivalent to the best floated gyros [37]. However, they are simpler, rugged, and less prone to failures.

Other typologies of gyroscopes are based on direct rate sensing on viscous coupling between fluids and masses [32]; vibration motion deflection due to Coriolis apparent forces, where a vibration element (e.g., a string, a hollow cylinder, a rod, a tuning fork, a hemispherical dome) is present to sense rotation-induced deflections; optical interferometry and Sagnac effect. Coriolis vibratory gyros, especially in the tuning fork version, are fundamental for the size reduction of gyroscopes, and the realization of silicon solid-state or MEMS sensors. They will be discussed in the following. Optical gyroscopes allow extremely high performance and accuracy since they measure optical length differences within a circular waveguide. In fact, when light travels in opposite directions (clockwise and anticlockwise) around an enclosed ring, differences arise in the apparent optical length of the two paths when the ring is rotated about an axis perpendicular to the plane containing the ring itself. The path differences can be practically detected as a phase shift, but, even in this case, the angular differences are extremely small [23]. To overcome this practical problem, interferometric fiber optic gyros and ring laser gyros have been ideated. The former use many turns of optical fiber to magnify the phase shift difference, the latter replace the phase measurement by a frequency difference measurement between two counterpropagating resonant laser beams. Despite their potential, optical sensors are not widespread in spacecraft applications since they require maintenance to overcome optical degradation, they are power consuming, they lose detection capability for very slow rotation rates, and they have a complex implementation to achieve good performance.

Recent technology developments allowed simpler, smaller, lighter, lower power consuming, and lower cost inertial sensors. These features

are all positive in spacecraft applications, but they become fundamentals for small satellites and CubeSats (cfr. section CubeSats in [Chapter 15](#) — Modern Spacecraft GNC), and they are possible, thanks to MEMS sensors [38,39]. MEMS sensors are packaged similarly to other integrated circuits, with a single part containing inertial sensors for multiple axes. They are produced with chemical etching and batch processing techniques, and they were recently qualified for operations in the space radiation environment. The major drawbacks of MEMS sensors are related with their short lifetime and reduced performance, compared to other inertial sensors. In fact, size reduction is commonly related to a decrease in sensitivity and an increase in noise. MEMS sensors are made of silicon wafers or piezoelectric quartz layers, and they are based on the same physical principles of the classical inertial sensors. MEMS gyroscopes exploit the Coriolis acceleration effect on vibrating proof masses to detect angular rotation, while MEMS accelerometers exploit miniaturized pendulum displacements or microresonators vibration frequency changes to detect linear acceleration. An array of MEMS sensors may be integrated into a single chip to provide multiple independent measurements of inertial motion. Moreover, single-chip implementations of six-axis, or nine-axis, IMUs have become common, guaranteeing a vast reduction in volume, negligible power consumption, and the ability to carry out a complete characterization of the unit in a single operation.

Typical error sources

Every inertial sensor is subject to errors which limit their measurement accuracy. The applications of these instruments in spacecraft shall consider their common error sources in order to minimize their impact on the GNC functions. Moreover, a proper GNC design shall rely on a proper sensor modeling that considers all the relevant errors affecting the measurements. The error sources of inertial sensors can be categorized in four classes:

- Deterministic (or constant) errors, which are repeatable. They can be determined and corrected by calibration.
- Temperature-induced errors, which can be determined and corrected by thermal calibration.
- Switch-on to switch-on errors, which are stochastic processes but stays constant for any single run. They can be corrected with on-board calibration estimation techniques, discussed in [Chapter 14](#) — Applicative GNC Cases and Examples.

- In-run stochastic variations errors, which should be modeled as stochastic processes varying throughout the measurement run. In-run stochastic variations with slow dynamics can be corrected with on-board calibration estimation techniques. Stochastic errors with a fast dynamics (i.e., random noise) cannot be corrected.

Typical error sources depend on the specific type of sensor. However, the major ones are in common to all of them.

Bias errors are related to a nonnull sensor output, which is present even in the absence of an applied input. They are typically expressed in units of degrees per hour ($^{\circ}/\text{h}$) for gyroscopes, and milli-g or micro-g (mg, μg) for accelerometers.

Bias errors in both gyroscopes and accelerometers can be further broken down into more error components:

- *Constant bias*: fixed bias term in every measurement of the sensor. It can be corrected by calibration of the sensor.
- *Bias repeatability* (switch-on to switch on bias): bias term varying at every sensor powerup. It remains constant until next switch-on. It can be corrected by in-run calibration with bias estimation techniques. A high bias repeatability guarantees a faster convergence of the estimation filters between distinct runs of the sensor.
- *Temperature-dependent bias*: bias term depending on the operating temperature of the sensor within the specified temperature range. It can be corrected by implementing a thermal model of the sensor and taking into account temperature measurements of the instrument.

Mechanical gyroscopes are also affected by *acceleration-dependent bias* (i.e., g-dependent bias), which is proportional to the magnitude of the applied acceleration. Such errors arise because of mass unbalance and asymmetry, due to imperfections caused by the fabrication process. They are expressed by means of coefficients having units of degrees per hour per g ($^{\circ}/\text{h/g}$).

Bias instability errors are related to an in-run bias drift with stochastic evolution, which can be characterized in terms of variance and process time. Bias instability has an impact on the long-term integration of sensor's outputs. It is expressed in units of degrees per hour ($^{\circ}/\text{h}$) for gyroscopes, and micro-g (μg) for accelerometers. It is defined with the AVAR method, which is introduced in the following dedicated section. Bias instability determines continuous fluctuations in the bias offset, which can be estimated on-board to improve motion propagation performance.

Random walk errors are pure stochastic processes associated to the characteristic of the intrinsic noise of the sensor. Integration of the random walk

noises in the measurements leads to a random walk in the final solution. They can be estimated by the AVAR method, which is introduced in (cfr. [Section AVAR](#)). Noise errors are defined according to the noise variance trend with respect to the averaging time.

- *Angle random walk (ARW)/Velocity random walk (VRW) errors* are high-frequency noises, and they can be observed as the short-term variations in the output. ARW is defined for gyroscopes, while VRW is defined for accelerometers. Integrating the sensor outputs, these terms cause random errors in angle/velocity estimation with a distribution that is proportional to the square root of the elapsed time. Thus, these errors will increase the longer integration and provide a fundamental limitation, together with bias instability, to any angle/velocity measurement that relies only on inertial sensors. They are expressed in units of degrees/meters per second per square root of hour ($^{\circ}/\sqrt{h}$, m/s/ \sqrt{h} or $\mu g/\sqrt{Hz}$). Note that, VRW can be also expressed in units of micro-g per square root of Hertz.
- *Rate random walk (RRW) errors* are long-term changes to the bias offset, which are randomly distributed over very long time. The time scale over which these changes occur can be defined by the RRW and allow for on-board recalibration of the sensor.

Random walk errors are classified in terms of characteristic time, and they are further differentiated from those associated with bias instability because of the different temporal properties, as analyzed in the power spectral density of the sensor stochastic processes.

Scale factor errors affect the ratio relating the change in the output signal to a change in the input to be measured. It is the measure of the gradient of the best straight line that can be fitted by the method of least squares to the expected sensor output signal, against input inertial signal over the full dynamic range at ambient temperature. Scale factor errors are commonly expressed as a ratio of output error to input quantity, in parts per million (ppm) or percentage. Additional errors arise from the nonlinearity of the scale factor with respect to the least squares linear fitting, and from its asymmetry with respect to opposite direction inputs.

Scale factor ratios are typically not constant and may fluctuate during in-run operations. *Scale factor instability* is an additional source of errors.

Cross-coupling errors determine spurious outputs resulting from sensor sensitivity to accelerations or angular rates about axes orthogonal to the input axis. Such errors arise through nonorthogonality of the sensor axes.

They are expressed as parts per mission (ppm) of the applied input. They can be corrected by calibration being repeatable deterministic errors.

Misalignment errors are related to a mechanical mounting error of the sensor triad with respect to the nominal body frame. The sensor outputs have couplings with respect to different body axes inputs. They are expressed in units of milliradians (mrad), and they are difficult to be distinguished with respect to cross-coupling errors. They can be corrected by calibration being repeatable deterministic errors.

Other than the previous sources of error, inertial sensors are characterized by other limitations, such as:

Input range limits associated to the maximum input values that can be meaningfully measured.

Output quantization associated to the mapping of a continuous signal into a discrete one and dependent on the analog to digital conversion resolution.

Latency associated to the sensor processing time, which determines a nonzero elapsed time between the input sensing and its output generation.

Moreover, inertial sensors are also affected by extrinsic errors due to the estimation processes elaborating inertial sensors outputs, as seen in [Fig. 6.25](#). For instance, an imprecise model of the gravitation field may lead to motion propagation errors. Similarly, numerical computation errors may also affect the precision of propagated position, velocity, or attitude vectors. However, these latter errors are associated to the navigation functions and not directly to the inertial sensor itself. [Fig. 6.26](#) summarizes the main sources of errors described in this section.

Inertial sensors performances

Inertial sensors are classified into various performance grades according to their typical accuracy levels, which are divided in four main performance categories:

- Consumer grade.
- Industrial grade.
- Tactical grade.
- Navigation grade.

These categories are mainly subdivided according to the in-run bias stability of the sensor, as the in-run bias stability plays such a large role in determining inertial navigation performance. In fact, a low value of the bias drift enables the application of inertial sensors in high-end market segments. The performance grades, associated to the available sensor technologies, are

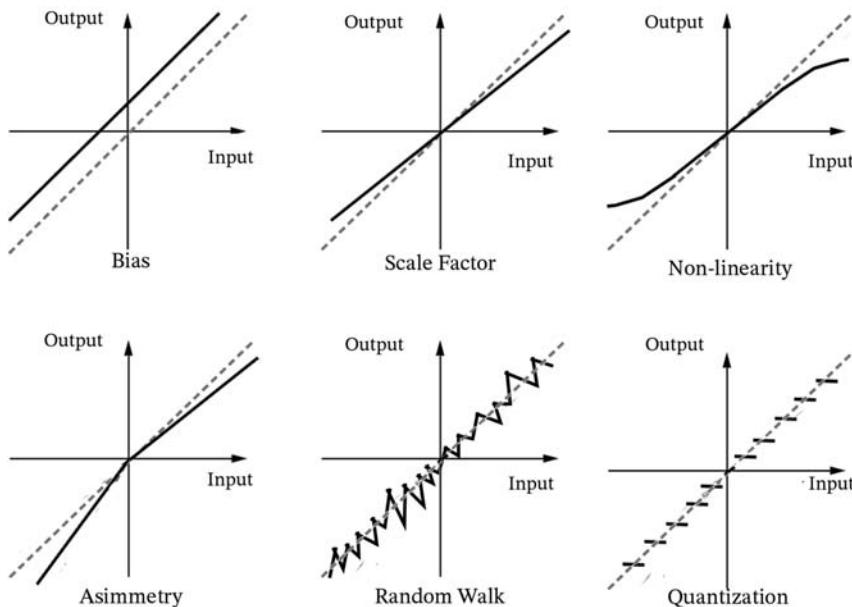


Figure 6.26 Inertial sensor errors.

reported in Figs. 6.27 and 6.28 [40,41]. Given the different performance of the sensors, the order of magnitude for position and angle accuracy estimation, after 1h of motion propagation, is reported in Table 6.11.

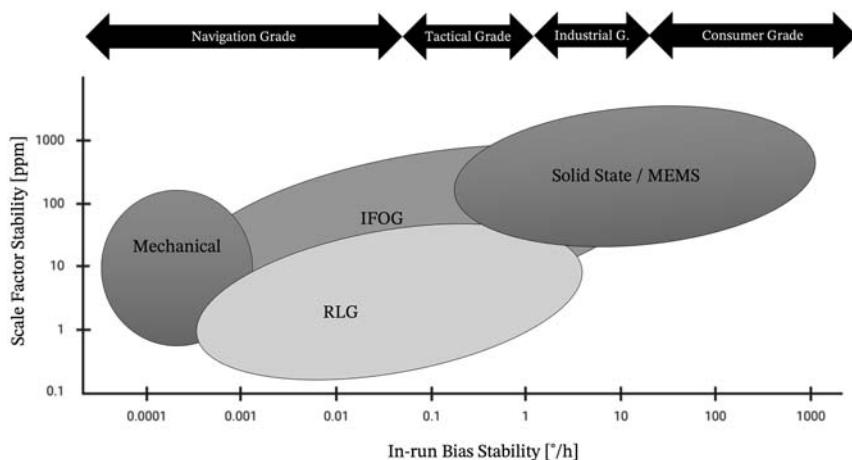


Figure 6.27 Gyroscope technologies and performance grades.

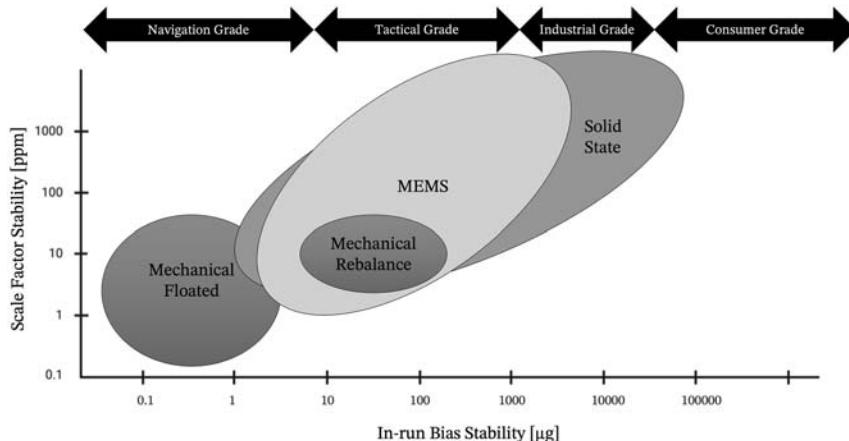


Figure 6.28 Accelerometer technologies and performance grades.

Table 6.11 Sensor grades and navigation performance after 1h of propagation.

Sensor grade	Angle accuracy ($^{\circ}$)	Position accuracy (km)
Consumer grade	>50	>1000
Industrial	5–50	100–1000
Tactical	0.1–5	1–100
Navigation	<0.1	<1

Allan variance and statistical error representation

Inertial sensors are affected by both repeatable deterministic errors and stochastic random errors. The former can be eliminated or reduced, thanks to standard sensor calibration techniques, while the latter are more difficult to be compensated, unless complex statistical estimation techniques are used. However, in general, it is almost impossible to completely remove their impact on the motion propagation performance.

In-run stochastic variations are due to temperature, aging, or mechanical stress on the system. Furthermore, electronic noise, interactions with the external environment, and transduction interferences are additional issues generating random errors on inertial sensors output. Thus, given this broad range of physical phenomena, stochastic error sources have different characteristic times, filling a complete frequency spectrum. In fact, they appear on the output as a noise or a slow change of parameters in time. The contribution of these errors at a specific time cannot be precisely predicted, but the analysis of these stochastic processes in inertial sensors is possible. According

to the international IEEE standard 952–1997 [42], two analysis methods are available, but the AVAR method is the most widely used. AVAR is the method of analysis of stochastic processes in a time domain, and it describes the intrinsic variance of a signal as a function of averaging time. It can evaluate the real performance of inertial sensors in relation to the effects of bias instability, ARW and RRW. Note that, often the AVAR term is also used to refer to its square root.

To compute the AVAR, we have to follow the following steps [40]:

1. *Data collection*: sample the output data, for a steady-state input condition, from the sensor at interval T_s to obtain a long sequence of consecutive N -point data.
2. *Data clustering*: divide the N -point data into K clusters, with each cluster having the length of M -point data.
3. *Data averaging*: average the data in each K -th cluster.

$$\bar{a}_K(M) = \frac{1}{M} \sum_{i=1}^M a_K(i)$$

4. *Calculate variance*: take the difference in average between successive clusters. There must be enough data for at least nine bins, otherwise the results obtained begin to lose their significance. The AVAR is defined as the mean value of the square of the difference of adjacent time averages from a time series as a function of averaging time, multiple of the sample time, $\tau = mT_s$, and it is expressed as:

$$\sigma_{\text{Allan}}^2(\tau) = \frac{1}{2(K-1)} \sum_{K=1}^{K-1} [\bar{a}_{K+1}(M) - \bar{a}_K(M)]^2$$

A typical Allan plot for inertial sensors is shown in Fig. 6.29, in which the values of the root AVAR are plotted against the averaging interval, τ , in log-log form. Different types of random processes cause slopes with different gradients to appear on the plot, and they also are correlated with different averaging times, and therefore correspond to different locations on the Allan plot. The basic noise terms are ARW, RRW, bias instability, quantization noise, and drift rate ramp. In addition, the sinusoidal noise and exponentially correlated noise can also be identified on the plot.

As shown in Fig. 6.29, the electronic quantization white noise is characterized on the AVAR plot as a slope with gradient equal to -1 . The magnitude of this noise can be read off the slope line at $\tau = \sqrt{3}$. This noise is strictly due to the digital nature of the sensors. The random walk

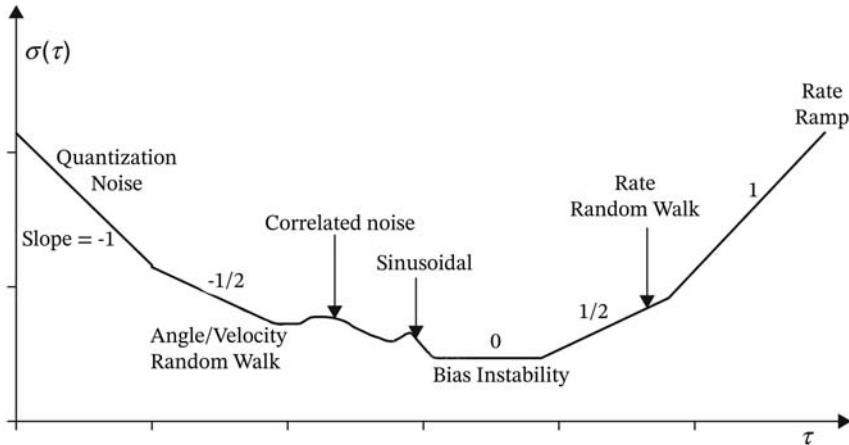


Figure 6.29 Example Allan variance analysis results adapted from [42].

measurement for sensor noise, such as the ARW for gyros and VRW for accelerometers, can be obtained directly by reading the slope line at $\tau = 1$. These noise terms are all characterized by a white noise spectrum on the output. Bias instability is due to low frequency bias fluctuations, which are characterized by $1/f$ noise or flicker noise spectrum, where f is the cutoff frequency of the sensor output. Bias instability appears on the plot as a plateau around the minimum. The flat region of the plot can be examined to estimate the limit of the bias instability, as well as the cutoff frequency of the underlying flicker noise. The RRW indicates a random walk with a very long correlation time. It is represented on the AVAR plot by a slope of $+1/2$. The magnitude of this noise, K , can be read off the slope line at $\tau = 3$. For long, but finite time intervals, this is more of a deterministic error rather than a random noise. Its presence in the data may indicate a very slow monotonic change in the sensor properties, and its magnitude can be obtained from the slope line at $\tau = \sqrt{2}$.

If it can be assumed that the existing random processes are all statistically independent, then it can be shown that the Allan variance at any given t is the sum of Allan variances due to the individual random processes at the same τ :

$$\sigma_{Allan}^2(\tau) = \sigma_{ARW}^2(\tau) + \sigma_{Quant}^2(\tau) + \sigma_{BiasInst}^2(\tau) + \sigma_{RRW}^2(\tau) + \dots$$

Thus, estimating the amplitude of a given random noise in any region of the AVAR requires a knowledge of the amplitudes of the other random

noises in the same region. Fig. 6.30 reports the individual AVAR contributions of the main random processes discussed above.

Note that, the computation of AVAR needs a finite number of clusters that can be generated from the raw data measurements of the sensors. Depending on the size of these clusters, the AVAR can identify any noise term that is affecting the data sensor. It is important to mention that the estimation accuracy of the AVAR for a given t depends on the number of independent clusters within the data set. The bigger the number of independent clusters, the better the estimation accuracy.

All the stochastic errors contribute to accumulate errors, with an increasing standard deviation, along the motion propagation. If a continuous bias estimation were possible, the propagation errors would theoretically approach zero, in absence of other noncalibrated errors. However, this is not feasible, and finite time periodic calibrations are the best viable solution. The frequency of these in-run on-board calibrations can be selected from the AVAR analyses. In fact, calibrating at least before the bias stability characteristic time makes ARW dominant and drift negligible. Given the time interval at which we can effectively recalibrate, we find out from the AVAR graph the statistical estimate of the propagation error. For example,

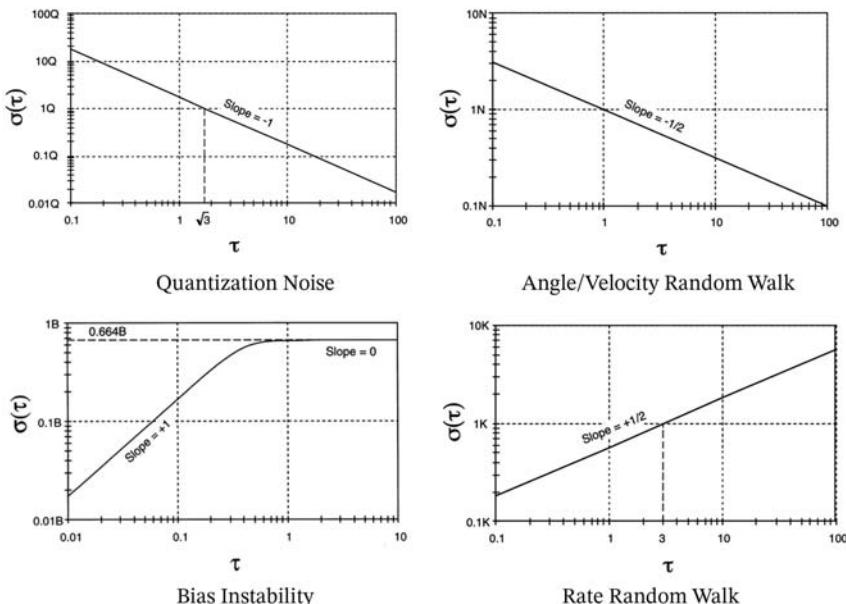


Figure 6.30 Individual Allan variance contributions adapted from Ref. [42].

assuming the root of the AVAR of a tactical grade MEMS gyroscope at $\tau = 2700\text{s}$ being $\sigma_{\text{Allan}} = 1.3^\circ/\text{h} = 361 \mu^\circ/\text{s}$, and propagating the motion along a $45\text{ min} = 2700\text{ s}$ window (e.g., an LEO eclipse period), the statistical propagation error would be $\epsilon_0 = 0.9747 \deg_{\text{RMS}}$.

As said, two main parameters represent the quality of inertial sensors. They are the ARW and the bias for gyroscopes, and the VRW and bias for accelerometers. Noting that bias and scale-factor drifts are much larger than noise contributions, Table 6.12 reports random walk ad bias values for typical inertial sensors.

Remind that inertial sensor units are note standardized, and different conversion factors are very useful to be known. Working with the square root of time, converting between hours and seconds is a factor of 60 instead of 3600: $60\sqrt{\text{s}} = \sqrt{\text{h}}$. Similarly, accelerations can be expressed in μg or in the SI $\text{m}/\text{s}/\text{s}$, and $1\mu\text{g} \approx 10^{-5}\text{m}/\text{s}/\text{s}$. Thus, for example, a VRW of $X \mu\text{g}/\sqrt{\text{Hz}} \approx X 6 \times 10^{-4}\text{m}/\text{s}/\sqrt{\text{h}}$.

Gyroscope model

A general gyroscope model including all the relevant sensor's errors is:

$$\boldsymbol{\omega}(t) = (\mathbf{I}_3 + \mathbf{SF}) \mathbf{OR} \boldsymbol{\omega}_{\text{true}}(t) + \mathbf{b}(t) + \boldsymbol{\eta}(t).$$

An example of this model implementation in MATLAB/Simulink is shown in Fig. 6.31. A similar structure can be extended of other sensor models, such as accelerometers or magnetometers. The implementation of the individual error sources would be the same in all the sensor models affected by the same errors. Note that bias and noise terms can also be

Table 6.12 Typical random walk noise and bias values for inertial sensors.

Sensor grade	Gyro bias (°/h)	ARW noise (°/ $\sqrt{\text{h}}$)	Acc bias (μg)	VRW noise ($\mu\text{g}/\sqrt{\text{Hz}}$)
Consumer grade	>50	>0.5	>50,000	>100
Industrial	5–50	0.1–0.5	1000–50,000	10–100
Tactical	0.1–5	0.05–0.1	10–1000	1–10
Navigation	<0.1	<0.05	<10	<1

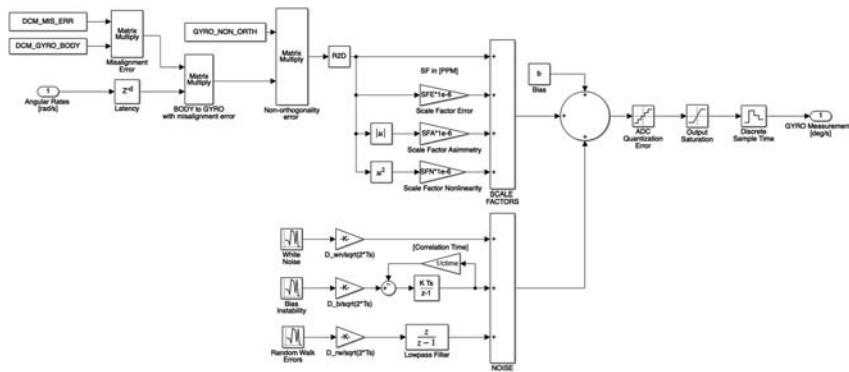


Figure 6.31 Gyroscope model.

multiplied by the scale factors, misalignment, and nonorthogonality errors (i.e., $(\mathbf{I}_3 + \mathbf{SF})\mathbf{OR}$). It is just a matter of their definition and implementation. In Chapter 14—Applicative GNC Cases and Examples, the other formulation is used to simplify the sensor calibration equations.

Electro-optical sensors

For unmanned missions, spacecraft GNC requires a level of autonomy that can only be provided, thanks to dedicated on-board GNC including electro-optical sensor suites. In fact, the measurements taken by this kind of sensors enable autonomous navigation, providing on-board data that are fundamental to have the spacecraft estimating its state.

To achieve the high level of autonomy and reliability required in today's space missions, navigation making use of high-accuracy measurements, to be processed on-board at high frequency, is mandatory in critical operations. The mission concept and the navigation technique will drive the selection of different sensor suites.

In general, the term electro-optical indicates all the instruments capable of collecting reflected or emitted radiations in a certain range of the electro-magnetic spectrum. In modern spacecraft applications, almost exclusively visible (0.37–0.75 μm) and infrared (0.75–1000 μm) bands are used, and the most common sensors are cameras (monocular and stereo) or LIDAR systems.

Cameras

Cameras are based on passive electro-optical sensors that collect incoming light waves (visible or infrared) and traduce them into signals which allow to produce an image. The two main types of electro-optical sensors employed by modern cameras are the charge-coupled device (CCD) and the active-pixel sensor (complementary metal oxide semiconductor (CMOS)). Without entering into technical details, both CCD and CMOS are composed by arrays of silicon photosites, also known as pixels, that convert collected photons into electrical signal. In a CCD sensor, each photosite is an analog device and the charge is transported across the chip and read at one corner of the array. An AD converter is used to convert the analog signal into a digital value. On the other hand, CMOS sensors rely on one or more transistors at each photosite that amplify and move the charge using more traditional wires. The main difference in terms of performance can be summarized here:

- CCD sensors are superior to CMOS for what concerns high-quality, low-noise images.
- Light sensitivity of a CMOS chip tends to be lower, as many of the photons hit the transistors instead of the photosite.
- CCD sensors require much higher power than an equivalent CMOS sensor (up to 100 times more).

Another important classification while referring to imaging sensors is the type of shutter adopted. The shutter is the device (mechanical or electric) allowing light to hit the sensor for a given amount of time (i.e., exposure time). Two main types of shutters can be identified: rolling and global. Rolling shutters expose sensor rows sequentially from top to bottom, while for global shutters, all the pixels of the sensors are exposed at the same time. For fast dynamics navigation scenario, it is generally preferred global (snapshot) shutter mode as rolling mode introduces many artifacts shifting in the image (smearing effect), and global shutter avoids preprocessing compensation. When the camera is moving very fast or spinning with respect to the target, with a roller shutter, the image is distorted because different lines can capture different times of the scene, generating a “moving picture.” One classical example of this undesired phenomenon is obtained while imaging the rotor of a helicopter. The rotor blades would appear to be unnaturally swept back due to the rolling shutter.

The optical lens (or a group of lenses) will complete the camera assembly. A lens is characterized by a set of two main parameters:

- Focal length: it determines the magnification of the image projection onto the image plane. It is also inversely proportional to the camera FOV, i.e., longer focal length will correspond to narrower FOV.

- Aperture: it determines the amount of light hitting the camera sensor.

Finally, a very important figure of merit, especially for space applications is the signal-to-noise ratio (SNR). The SNR represents a quantitative value describing the quality of a measurement. As the name suggests, SNR is obtained as a ratio between the measured signal and the overall measured noise. High values of noise imply very good measurement of the target object, i.e., the incoming signal is very strong with respect to the overall noise. On the contrary, low SNR can produce invalid measurements and consequent target undetectability. This is the case, for example, of very far objects having a very low signal measured by the sensor. In fact, the incoming signal directly depends on the photon flux hitting the sensor (therefore on the integration time, sensor quantum efficiency, and lens optical transmittance). On the other hand, the sources of noise are almost exclusively dependent on the physical characteristics of the sensor. The main noise sources are summarized in [Table 6.13](#).

Table 6.13 Typical noise sources.

Noise	Description	Unit
Read noise	Inherent to the process of converting charge carriers into a voltage signal for quantification, and analog-to-digital conversion.	e^-
Dark current	Represents the level of thermally generated electrons, directly dependent on exposure time and temperature.	$\frac{e^-}{s}$
Signal shot noise	Inherent natural variation of the incident photon flux.	$\frac{e^-}{s}$
Sky/background noise	Overall optical noise of the system, composed by the electronic noise in the sensor, and undesired stray light.	$\frac{e^-}{s}$

Applicability

Camera systems have the advantage of being inexpensive, passive, compact, and power efficient. Furthermore, they provide images at high frame rates that are readily interpretable by human operators for improved situational awareness. Cameras are typically usable at ranges of a few centimeters up to around 100 m, with the exact range being determined by the interplay among the camera imaging resolution, field of view size, physical dimensions of visual targets, and the employed features (i.e., centroid, shape, reference points). On the downside, their image output should be processed by complex algorithms for extracting cues guiding the estimation of spacecraft state. These algorithms can have low robustness and are often computationally demanding. Camera-based approaches should also deal with foreground segmentation in order to isolate the relevant sensory input from its potentially confusing background. Furthermore, to operate in low light conditions, camera sensors have to depend on light emitted from external sources. Thermal cameras have also been proposed to avoid illumination problems or exploited as aide to the main optical system even though they offer much lower accuracy [43]. Approaches using a single camera are referred to as monocular. However, the low cost of camera units permits pose estimation using two or even more such sensors, giving rise to approaches referred to as binocular or multiocular, respectively. A list example of vision-based navigation scenarios may include planetary approach application, satellite servicing, active debris removal, Descent and Landing scenario (high or low dynamics), general rendezvous, and docking application or rovers' navigation (localization and mapping). In most of the missions, the navigation cameras will provide a gray-scale image in the visible wavelength. Depending on the FOV of the camera, we will refer to Narrow-Angle Cameras (NACs) or Wide-Angle Cameras (WACs). We refer to NAC when the optics design provides an FOV in the range of 1–10°. On some occasions, but is not very common, a Medium-Angle Camera (MAC) may be defined, with FOV in between 10 and 30°. Finally, a WAC usually has an FOV larger than 20°. NAC will be used when the target is very far from the observer. Having a smaller FOV implies to have a larger target apparent dimension on the camera sensor. This is useful at far range, but it can become problematic at close range when the target may occupy all the FOV dimension or high accuracy pointing is necessary to keep it in view. WAC will be used for medium to close distances. The closer the target is, the wider the FOV may be necessary to satisfy target object or surface visibility inside the FOV. A larger FOV will also be considered.

in certain scenarios where the target has fast dynamics and crosses the detector. In this case, a larger FOV allows to capture the streak of the target in one or more consecutive images granting navigation data extraction. We shall also note that wide FOV, as for instance 50° or 70° introduces high image distortion. The narrower the field of view, the larger the baffle design is needed. Baffle cone will protect the optics and image from light sources outside of the FOV, generating undesired stray lights. A WAC or NAC camera will normally offer not only images in visible wavelength but also in many cases, it also covers near-infrared. When the mission accomplishment will require operations in eclipse or poor illumination conditions, either active illuminance beam is needed or Thermal Infrared (TIR) cameras. Finally, different studies are also analyzing the utilization of Multispectral or Hyperspectral cameras for navigation [44]. Visible and TIR is the most convenient combination, while hyperspectral cameras may better facilitate a reused concept of instrument to be used both for navigation and for scientific purposes as characterization or different science observations. For rovers' navigation or for close operations where 3D information is required, stereovision cameras are selected. Stereo cameras are composed by two optical heads aligned, temporally synchronized and with certain distance separation granting overlap of the FOV. In certain missions, the images will only be acquired on-board while information extraction process is performed on ground, because of the on-board computing constraints. On-board autonomy is mandatory for critical operations, such as scenarios involving fast dynamics or huge delays and latency in communication that would make ground operations impossible. As a clear example, ground operations are impossible if the spacecraft GNC is dealing with descent and landing operations on a small body, which requires to extract data from images at the slowest frequency of 1 frame per second, and the spacecraft to or from ground communication takes 9 min. The on-board processing in that case is mandatory, under a big level of autonomy, and it requires fast image acquisition by navigation cameras, as well as on-board camera image corrections (preprocessing), before the data extraction. Ground operations may acquire raw images and perform the image preprocessing correction offline to have a more accurate output, while for on-board processing, these corrections should be applied on the fly. The image preprocessing corrections include radiometric and geometric calibration. Radiometric calibration is performed to produce an image that ideally would correspond to a perfect detector while the geometric calibration corrects distortions produced by the finite-aperture optics and motion. Optionally, it is also possible to

validate the images in order to detect (and discard) “contaminated” images that may degrade the optical navigation performance. In fact, processing only the best shots improve both the navigation accuracy and the robustness. The low-level image correction functionalities should be, for example, in charge, but not limited to, of the defective pixel correction, photoresponse nonuniformity correction, dark signal correction, background noise reduction due to ionizing particle events or others, motion or aberration correction, and the focal length scale factor calibration. The preprocessing functions should be considered in the sensor suite selection. The navigation cameras should also provide certain metadata together with the images, for the proper identification and relevant information used in the navigation processing. These metadata include health status, self-tests, configuration options, thermal measurements, and very important, a time stamp of the image. There are several camera operation parameters that should be configurable for the proper operation, such as integration time, window position and size, start of integration, and acquisition options as binning, cropping, or even image stacking (stacking improves the SNR and help faint objects detection in trade-off configuration with integration time). The resolution in pixels vary and should be traded-off in between the need for large amount of information and the computational effort that would require processing it. Typical pixel resolutions trade-off selection uses images of 2048×2048 or 1024×1024 pixels. Among the most interesting parameter to consider in the camera selection is the pixel size, image area, frame rate, quantum efficiency, noise, modulation transfer function (MTF), environmental endurance, and shutter property.

A navigation camera for space shall provide good SNR (high sensitivity, low noise), fast frame rates, high pixel count, and low power consumption, as well as being cost-effective production. Both the rolling shutter artifacts as well as the global shutter drawbacks can be compensated, but this means adding logic and complexity, translated in cost.

Design

A navigation camera design shall encompass the following steps:

- Image sensors selection. The sensor size has to be selected according to the required accuracy, considering the mission scenario and desired navigation performance.

- SNR analysis and magnitude detectability. Given the target characteristics and the mission scenarios, an evaluation of the object magnitude and of the resulting SNR is necessary to guarantee observability.
- Optical design. It is the process of selecting and assembling optical components to satisfy the lens requirements. This step is necessary when commercial lenses are not available for the desired application.
- Stray light assessment and baffle design. Light sources (e.g., Sun, planets) can generate undesired stray light hitting the sensor, degrading imaging, and consequently, navigation performance. For this reason, the stray light sources should be identified for the given mission profile and a proper baffle should be designed to limit this undesired effect. In fact, baffles are mechanical systems that, through a series of internal reflections, can shield the incoming stray lights.
- Thermal design. Camera working domain must be ensured through proper thermal regulation systems to guarantee expected performance.

Additional mechanical features as pointing structures may not be directly related to the camera but to the assembly in the satellite, either under the responsibility of the satellite bus platform or under the GNC or AOCS responsible.

Table 6.14 summarizes the strength and weakness of cameras:

LIDAR

LIDAR is a type of electro-optical sensor that use beam-steering devices to actively direct a laser beam in particular directions and rely on different physical principles (namely triangulation, phase shift, or time-of-flight) to measure range. An example of the working principle of scanning LIDARs is represented in [Fig. 6.32](#).

Table 6.14 Cameras strengths and weaknesses.

Strengths	Weaknesses
Reliable	Highly sensible to illumination conditions
Affordable	Not robust to the presence of celestial body in the field of view
Low power consumption Low hardware complexity Can be used for supervised applications	

The primary advantage of laser-based sensors is that they offer highly accurate output, which is often combined with a large operating range. Depending on their operating principle, their operating range is between a few meters (triangulation) to several kilometers (time-of-flight). The operating principle also influences the sensor speed, i.e., the number of point measurements that can be accumulated per second. Besides taking longer to complete a scan, a low scanning speed also implies that measurements can be distorted due to the relative motion between the sensor and the target during the scan. This, however, is overcome by the recently introduced flash (or detector array) LIDARs, which achieve high frame rates by emitting light pulses within their entire field of view, bypassing scanning. The working principle for this kind of LIDARs is depicted in Fig. 6.33.

Laser scanners readily provide 3D measurements which are also insensitive to illumination changes and shadows. Their primary drawbacks are that they are expensive, bulkier, less power efficient, and have a lower resolution and a small field of view. Moreover, they imply higher computational cost to process a bigger amount of data, usually point clouds. These drawbacks are particularly true for flash LIDARs, which require many times more instantaneous power than scanning LIDARs because they need to divide the power across many detectors of the pixel array, but they don't have any moving parts. This division limits practical combinations of their range, FOV size, and resolution. As a general guideline, flash LIDARS are better suited to scenarios involving faster objects within a smaller volume. It is also noted here that a small field of view might in turn constrain a chaser's approach trajectory toward a target spacecraft. A summary of the main advantages and drawbacks is reported in Table 6.15.

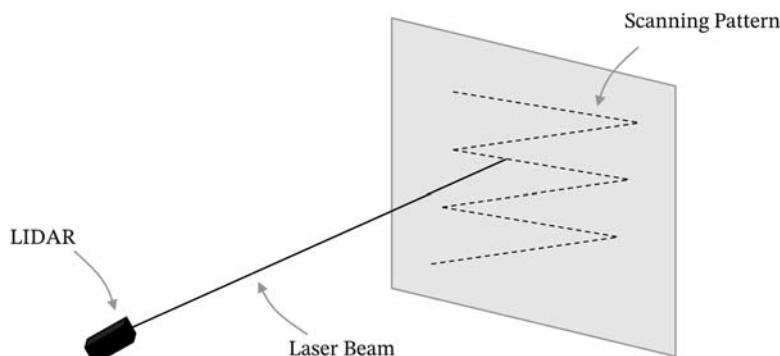


Figure 6.32 Scanning LIDAR.

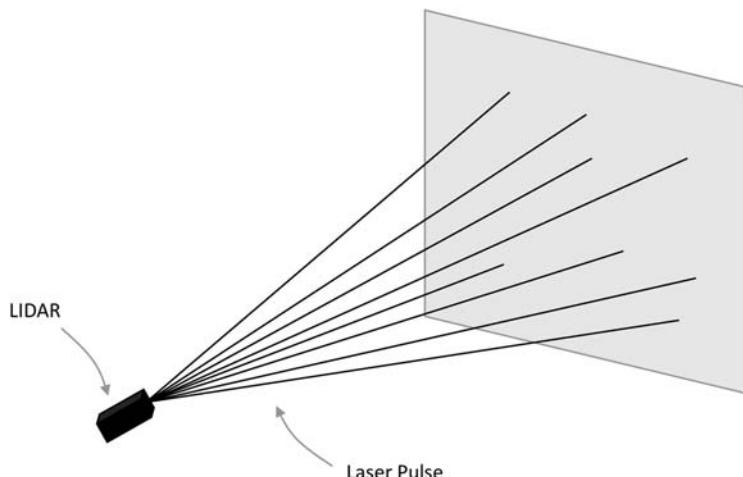


Figure 6.33 Flash LIDAR.

Table 6.15 LIDAR strengths and weaknesses.

Strengths	Weaknesses
Robust to operations with poor illumination conditions	High cost
Easy segmentation	High power consumption
3D position measurements	Complex hardware High computational cost

Altimeters

Satellite altimetry was developed in the early 60s, and it has generally been used to provide accurate measurements of the shape of a planetary surface. This kind of measurement can be widely applied to oceanography, geodesy, and geophysics. The altimeter working principle is to measure the travel time of an electromagnetic signal from the spacecraft to the planetary surface and back. This is common to all different kinds of altimeters, but the properties of the signal can infer its class: single beam, multibeam, or scanning synthetic aperture system. Furthermore, the operating frequency of the instrument implies a further classification, i.e., radar and laser. For science applications, scanning and multibeam systems are widely used. On the

contrary, in the GNC domain, mostly single beam altimeters are adopted. In this chapter, the main characteristics of single beam radar and laser altimeters are presented.

Altimetry principles

The objective of satellite altimetry is to measure the range between the satellite and the planet surface. A primary design parameter of an altimeter system is the area on the planet surface over which the range from the altimeter to the reference surface is measured. Intuitively, this can be approximated as the antenna footprint. The footprint of an antenna is traditionally defined, in a *beam-limited* sense, as the area on the target surface within the field of view subtended by the beamwidth of the antenna gain pattern. Thus, narrow beams have a higher accuracy, but they would require very large antennae that are not viable for space missions. Moreover, this kind of functional mode is more sensible to mispointing. In a *pulse-limited* configuration, the same results can be achieved with an antenna with a smaller diameter. This is possible because the return pulse is dictated by the length of the emitting pulse as shown in Fig. 6.34. In other words, the radiated phase front meets the scattering surface over an area that is limited by the width of the transmitted pulse.

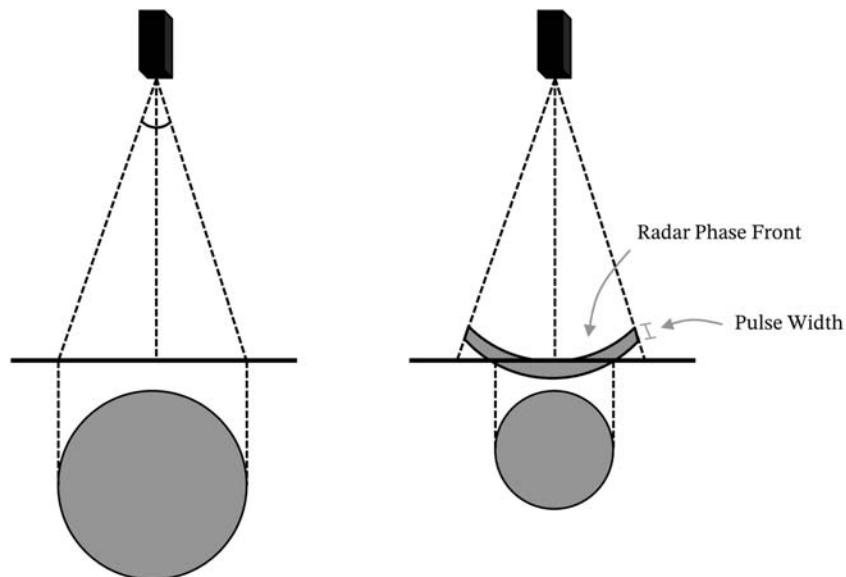


Figure 6.34 Beam-limited (left) and pulse-limited (right) footprint.

All the radar altimeters flown in space are pulse-limited, except for laser altimeters that are usually beam-limited due to their higher working frequency.

Radar and laser altimeters

The main difference between radar and laser altimeter is their working frequency. Both approaches have their strength and weakness that are summarized in [Table 6.16](#).

As previously detailed, radar altimeters have a larger footprint with respect to laser altimeters. This implies a lower “horizontal” resolution but a comparable “vertical” accuracy. It is important to consider that radar altimeters do not suffer of attenuation in the presence of atmosphere. This is usually relevant for Earth applications (laser altimeters are weather dependents), but it may be neglected for some space applications. Size and mass are comparable, but radar antennae are usually larger and heavier but with a limited power consumption.

While several past Moon and Mars missions relied on radar altimeters [[45,46](#)], the current trend is to move toward laser-based sensors, such as LiDARs [[47](#)]. Despite the kind of altimeters, some aspects directly related to GNC can be highlighted. An advantage of using altimeters as additional sensors is that altitude measurements can be continuously collected without sunlight or visibility restrictions. However, an altimeter provides measurements relative to the surface, which may have an unknown height above the planet’s center of mass; this is particularly true for small-body and unknown targets. In these cases, a shape model needs to be estimated or updated simultaneously with the spacecraft’s orbit during the initial approach phase. Note that altimeters are extremely useful in GNC for

Table 6.16 Radar versus laser altimeter.

	Radar altimeter	Laser altimeter
Footprint	km level	m level
Vertical accuracy	Tens of centimeters	Tens of centimeters
Atmospheric Attenuation	Yes	No
Size and mass	Higher	Lower
Power consumption	Lower	Higher

landing applications, since in this case, the relative distance with respect to the surface is the fundamental quantity to measure.

Altimeter model

A simple model of a single beam altimeter can be implemented by considering a single ray hitting a spherical surface of a planet or an astral body. A simplified ray-tracing algorithm can be employed to find the intersection between the altimeter beam and the spherical body's surface. Finding intersection points between a ray and a sphere is a known problem, and it can be tackled by solving a quadratic equation:

$$\mathbf{A}t^2 + \mathbf{B}t + \mathbf{C} = 0,$$

with:

$$\mathbf{A} = \mathbf{d} \cdot \mathbf{d}$$

$$\mathbf{B} = 2\mathbf{p} \cdot \mathbf{d}$$

$$\mathbf{C} = \mathbf{p} \cdot \mathbf{p} - R^2$$

where \mathbf{d} is the direction of the ray, \mathbf{p} is the vector from the origin of the ray to the center of the sphere, t is the length of the ray, and R is the radius of the body. This quadratic equation has two solutions, corresponding to the two points on the sphere as in Fig. 6.35.

Furthermore, the determinant of the quadratic equation ($\Delta = B^2 - 4AC$) helps identifying three different cases:

- $\Delta < 0$: there is no intersection with the sphere.
- $\Delta = 0$: there is only one solution of the equation, i.e., the ray is tangent to the sphere surface.
- $\Delta > 0$: there are two solutions corresponding to the two segments $\overline{SP_1}$ and $\overline{SP_2}$ with respect to the sensor position S.

The 3D points in the planet centered frame can be computed as:

$$\mathbf{P}_1 = \mathbf{p} + \mathbf{d} * \overline{SP_1}$$

$$\mathbf{P}_2 = \mathbf{p} + \mathbf{d} * \overline{SP_2}$$

In order to determine which point is actually observed, the projections of the two vectors onto the beam direction are compared and the smallest one is taken as observed point.

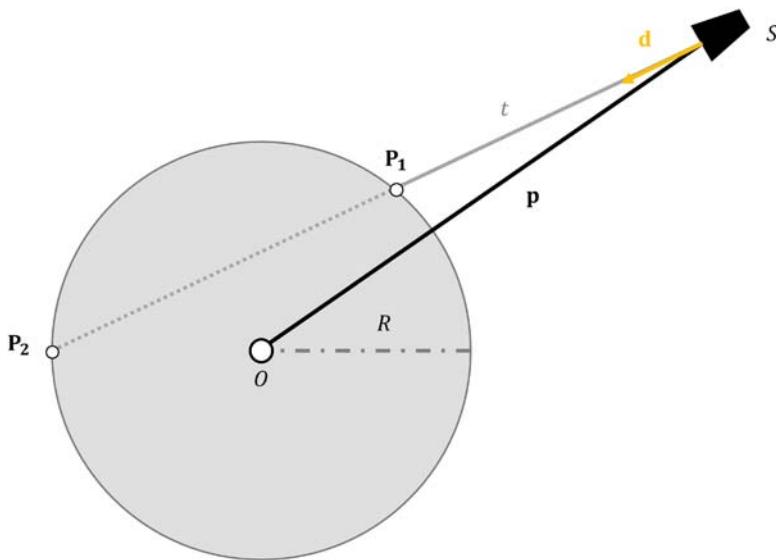


Figure 6.35 Ray-sphere intersection.

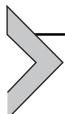
An alternative approach exploits a more complex model, considering the actual texture and roughness of the surface of the planet. This is done by performing a ray tracing with a 3D model of the observed body. This operation cannot be described by a simple mathematical model, and ad hoc solutions should be implemented.

References

- [1] Karris, S.T., Introduction to Simulink® with Engineering Applications, Orchard Publications.
- [2] International Vocabulary of Metrology—Basic and General Concepts and Associated Terms, JCGM, 2012. : <http://www.bipm.org/en/publications/guides/vim.html>.
- [3] J. Morrison, Statistics for Engineers: An Introduction, John Wiley & Sons, 2009.
- [4] W.J. DeCoursey, Statistics and Probability for Engineering Applications with Microsoft Excel, Newnes – Elsevier, 2003.
- [5] T. Forsberg, N. Grip, N. Sabourova, Non-Iterative Calibration for Accelerometers With Three Non-orthogonal Axes and Cross-Axis Interference, 2012.
- [6] S.U. Jan, Y.D. Lee, J. Shin, I. Koo, Sensor fault classification based on support vector machine and statistical time-domain features, IEEE Access 5 (2017) 8682–8690.
- [7] A. Colagrossi, M. Lavagna, Fault tolerant attitude and orbit determination system for small satellite platforms, Aerospace 9 (2022) 46, <https://doi.org/10.3390/aerospace9020046>.
- [8] <https://www.gps.gov/>.
- [9] https://www.glonass-iac.ru/en/about_glonass/.

- [10] <https://www.gsc-europa.eu/>.
- [11] <http://en.beidou.gov.cn/>.
- [12] <https://qzss.go.jp/en/>.
- [13] P.J. Teunissen, O. Montenbruck (Eds.), Springer Handbook of Global Navigation Satellite Systems, 10, Springer International Publishing, New York, NY, USA, 2017, pp. 978–983.
- [14] H. Liu, X. Cheng, G. Tang, J. Peng, GNSS performance research for MEO, GEO, and HEO, in: China Satellite Navigation Conference, Springer, Singapore, May 2017, pp. 37–45.
- [15] A. Delépaut, P. Giordano, J. Ventura-Traveset, D. Blonski, M. Schönfeldt, P. Schoonejans, R. Walker, Use of GNSS for lunar missions and plans for lunar in-orbit development, *Advances in Space Research* 66 (12) (2020) 2739–2756.
- [16] J. Miller, Planetary Spacecraft Navigation, Springer International Publishing, 2019.
- [17] D.W. Cirkendall, J.S. Border, Delta-DOR: the one-nanoradian navigation measurement system of the deep space network—history, architecture, and componentry, *The Interplanetary Network Progress Report* 42 (2013) 193.
- [18] Y. Doat, M. Lanucara, P.M. Besso, T. Beck, G. Lorenzo, M. Butkowic, ESA tracking network—A European asset, in: 2018 SpaceOps Conference, 2018, p. 2306.
- [19] J.D. Searcy, H.J. Pernicka, Magnetometer-only attitude determination using novel two-step Kalman filter approach, *Journal of Guidance, Control, and Dynamics* 35 (6) (2012) 1693–1701.
- [20] S. Carletta, P. Teofilatto, M. Salim Farissi, A magnetometer-only attitude determination strategy for small satellites: design of the algorithm and hardware-in-the-loop testing, *Aerospace* 7 (1) (2020) 3.
- [21] P. Tortora, Y. Oshman, F. Santoni, Spacecraft angular rate estimation from magnetometer data only using an analytic predictor, *Journal of Guidance, Control, and Dynamics* 27 (3) (2004) 365–373.
- [22] NASA, Space Vehicle Design Criteria: Spacecraft Sun Sensors, NASA SP-8047, 1970.
- [23] F.L. Markley, J.L. Crassidis, Fundamentals of Spacecraft Attitude Determination and Control, Space Technology Library, Springer, New York, NY, 2014.
- [24] B.O. Sunde (MSc. thesis), Sensor Modelling and Attitude Determination for Microsatellite, 90, Norwegian University of Science and Technology, 2005.
- [25] NASA, Space Vehicle Design Criteria: Spacecraft Earth Horizon Sensors, NASA SP-8033, 1969.
- [26] J.R. Wertz, Spacecraft Attitude Determination and Control, Kluwer Academic, Dordrecht, 1978.
- [27] European Cooperation for Space Standardization, ECSS-E-ST-60-20C Rev. 2 – Star Sensors Terminology and Performance Specification, ESA Requirements and Standards Division, 2019.
- [28] C.C. Liebe, Star trackers for attitude determination, *IEEE Aerospace and Electronic Systems Magazine* 10 (6) (1995) 10–16, <https://doi.org/10.1109/62.387971>.
- [29] D. Hoag, Apollo Guidance and Navigation Considerations of Apollo IMU Gimbal Lock, MIT Instrumentation Laboratory Document E-1344, 1963.
- [30] A.D. King, Inertial Navigation – Forty Years of Evolution, 1998.
- [31] T.F. Wiener, Theoretical Analysis of Gimballess Inertial Reference Equipment Using Delta-Modulated Instruments (MIT Ph.D. thesis), 1962.
- [32] D.H. Titterton, J.L. Weston, Strapdown Inertial Navigation Technology, second ed., The institution of Electrical Engineers, 2004.
- [33] D.F. Parker, G.A. Maugin, Recent Developments in Surface Acoustic Waves, Springer Verlag, 1988.

- [34] S. Middlehoek, S.A. Audet, *Silicon Sensors*, Academic Press, 1989.
- [35] M. Young, *Optics and Lasers*, Springer Verlag, 1992.
- [36] A.D. Kersey, T.A. Berkoff, W.W. Morsey, Fibre-grating based strain sensor with phase sensitive detection, in: *Proceedings 1st European Conference on Smart Structures and Materials*, 1992.
- [37] S. Merhav, *Aerospace Sensor Systems and Applications*, Springer, New York, 1962.
- [38] N. Barbour, Inertial navigation sensors, *NATO RTO lecture series-232, Advances in Navigation Sensors and Integration Technology* (2003).
- [39] N. Barbour, R. Anderson, J. Connelly, et al., Inertial MEMS system applications, *NATO RTO lecture series-232, Advances in Navigation Sensors and Integration Technology* (2003).
- [40] Y. Dong, MEMS inertial navigation systems for aircraft, in: *MEMS for Automotive and Aerospace Applications*, Woodhead Publishing, 2013.
- [41] G. Langfelder, M. Bestetti, M. Gadola, Silicon MEMS inertial sensors evolution over a quarter century, *Journal of Micromechanics and Microengineering* 31 (2021).
- [42] IEEE Standard Specification Format Guide and Test Procedure for Single-Axis Interferometric Fiber Optic Gyros, IEEE Std 952-1997, 1998.
- [43] G.B. Palmerini, Combining thermal and visual imaging in spacecraft proximity operations, in: *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, IEEE, December 2014, pp. 383–388.
- [44] D. Rondao, N. Aouf, M.A. Richardson, O. Dubois-Matra, Benchmarking of local feature detectors and descriptors for multispectral relative navigation in space, *Acta Astronautica* 172 (2020) 100–122.
- [45] B.D. Pollard, C.W. Chen, A radar terminal descent sensor for the mars science laboratory mission, in: *2009 IEEE Aerospace Conference*. IEEE, 2009.
- [46] R.D. Braun, R.M. Manning, Mars exploration entry, descent and landing challenges, in: *2006 IEEE Aerospace Conference*. IEEE, 2006.
- [47] T.P. Setterfield, et al., LiDAR-inertial Based Navigation and Mapping for Precision Landing, 2021.
- [48] ECSS-E-ST-60-20C Rev. 2, Stars Sensors Terminology and Performance Specification.
- [49] M.E. Pittelkau, Sensors for attitude determination, in: R. Blockley, W. Shyy (Eds.), *Encyclopedia of Aerospace Engineering*, Wiley, Chichester, 2010.



Actuators

**Andrea Colagrossi¹, Lisa Whittle², Vincenzo Pesce³,
Stefano Silvestrini¹, Matteo Battilana⁴**

¹Politecnico di Milano, Milan, Italy

²Asteroid Exploration, Leiden, the Netherlands

³Airbus D&S Advanced Studies, Toulouse, France

⁴OHB Italia S.p.A., Milan, Italy

Actuators are the ultimate elements of the guidance, navigation, and control (GNC) chain, executing forces and torques as commanded by the control section. They are the components physically injecting the output of the GNC into the dynamics in a way that the spacecraft can reach the desired reference state. With the usual human metaphor, actuators are the body parts allowing us to move around the world and to accomplish tasks (e.g., legs, arms, hands, etc.). The GNC design shall select and configure the actuators on the spacecraft, such that all the computed control forces and torques are correctly applied on the dynamics, both translational and rotational. For these reasons, the GNC designer shall know, understand, and correctly model the actuators to support its design process. This chapter is dedicated to briefly present all the major actuator typologies for spacecraft GNC applications, to discuss their operating principles, to highlight their peculiarities, strengths, and weaknesses, to categorize the available components according to their performances, and to discuss the most relevant modeling techniques and rules.

This chapter is structured as follows:

- *Actuator modeling for GNC.* In this section, the actuator modeling techniques are presented, discussing the main implementation principles of the most common numerical models to include the actuators behavior in the GNC design, analysis, verification, and validation phases. The principles of errors modeling and the most common actuators faults are also introduced.
- *Thrusters.* This section presents the main operating principles and the main component assemblies of thrusters for space applications, it introduces the concept of thrust management actuation functions, and it shows a simple thruster model implementation in MATLAB/Simulink.

- *Reaction wheels.* This section introduces the reaction wheels, and it presents the main characteristics of these actuation components. Friction and microvibrations affecting the dynamics of reaction wheels are briefly discussed, together with the actuation functions to operate multiple reaction wheels assemblies. An example reaction wheels model implemented in MATLAB/Simulink is also described.
- *Control moment gyros.* In this section, control moment gyros (CMGs) are presented and compared with reaction wheels as alternative angular momentum exchange devices. Their operating principles and functional characteristics are also discussed.
- *Magnetorquers.* This section describes the magnetic actuators for spacecraft applications, including their main limitations to guarantee a complete three-axis controllability. Alternative magnetorquers assemblies are presented, and a simple MATLAB/Simulink model implementation is shown at the end of the section.



Actuator modeling for GNC

Correct actuator modeling is fundamental for GNC applications. In fact, algorithms testing shall rely on proper actuator models to have a truthful representation of the available control forces and torques. Moreover, the control laws shall be designed and implemented to positively exploit the on-board actuation capabilities. Then, proper actuator models are needed to outline the best actuation functions and to minimize the impact of actuator's limitations on the GNC system. Consequently, we shall learn what the model of an actuator is, how it is built, and how it is used during the GNC design and verification processes. The following discussion will repeat, for reader's convenience, some concepts already introduced in the sensor modeling section of [Chapter 6](#) – Sensors, but it will underline those concepts that are peculiar for the actuator modeling.

An actuator is a component exploiting a physical principle to apply a force or a torque on the spacecraft while on orbit. Given the characteristics of the space environment, the methods to do so are limited with respect to what is available on the Earth. They are mainly based on momentum exchange and conservation principles, electromagnetic interactions, and fluid management.

The first method is by far the most used and relevant for space applications. Specifically, momentum exchange devices can put a mass in rotations to generate reaction torques, as in reaction wheels, or CMGs, but they can

also accelerate a mass along a certain direction, producing a force in the opposite one, as in the case of thrusters. Note that space propulsions and thrusters are based on Newton's third law of action-reaction, which has been used by Newton himself to derive the law of conservation of momentum. Moreover, thrusters can be even used to generate torques if they have an offset with respect to the Center of Mass (CoM) of the spacecraft.

Components based on electromagnetic interaction principles are commonly applied to generate a control torque on spacecraft orbiting around planets with a magnetic field, like in the case of magnetorquers. It shall be noted that electromagnetic thrusters exploit electromagnetism to accelerate ions to create a force. Hence, they belong to the class of momentum conservation actuators, despite their name might be misleading.

Finally, fluid management actuators can be used to dissipate spacecraft energy by means of internal viscous forces. However, their applications are very rare, and often limited just to nutation damping [1].

Some applications also exploited large external spacecraft surfaces to generate forces or torques by using solar radiation or aerodynamics pressure. The former is based on momentum exchange between the spacecraft and the electromagnetic radiation, the latter uses aerodynamic surfaces to actively control lift and drag values. However, both methods are not very common, and they have limited applicability range [2–5].

The actuator receives a command by the control block, after the actuation function processing, and it elaborates it to operate its internal component to generate the desired forces and torques. In modern spacecraft applications, the command interface is commonly digital, and the internal processing units convert the commands in analogic electrical signals for the final actuation (e.g., motor speed, valve opening status, and current intensity). During the whole process, the desired status is inevitably distorted and modified with respect to the commanded one. Moreover, internal frictions, electrical resistance, oscillations, vibrations, and other spurious effects further disturb the actuator's output. A proper actuator model shall be able to represent these deviations. We are going to mathematically refer to this as:

$$\overset{\circ}{\mathbf{f}}(t) = \mathbf{f}(t) + \boldsymbol{\varepsilon}_f(t), \quad (7.1)$$

$$\overset{\circ}{\mathbf{t}}(t) = \mathbf{t}(t) + \boldsymbol{\varepsilon}_t(t), \quad (7.2)$$

where $\check{\mathbf{f}}(t)$ and $\check{\mathbf{t}}(t)$ are the actuated force and torque, $\mathbf{f}(t)$ and $\mathbf{t}(t)$ are the quantities commanded by the GNC system, $\mathbf{e}_f(t)$ and $\mathbf{e}_t(t)$ are the respective overall errors and spurious effects.

An actuator model is a mathematic representation of the actuation process converting the desired command into the physical output generating the force or the torque. It is developed for a specific modeling purpose, and, in fact, there exist two main types of actuator models: functional and performance models.

Functional actuator models are implemented to serve as input for detailed analyses and accurate simulations. They describe each internal function and physical phenomena along the actuation processing, and they are representative of the actuator dynamics for realistic output profiles. The complete equipment elaboration chain is typically modeled, from command acquisition to internal element operations and dynamics. Functional models are used to accurately simulate the actuator dynamics to estimate the impact on the spacecraft motion. In addition, they allow to estimate if the actuator is correctly operated inside the prescribed ranges.

Performance actuator models represent the way in which the actuator generate forces and torques, including the main limitations and errors. Moreover, they quantify the consumed resources in order to support the system design process. Primarily, for what concern the power budget and the propellant mass budget. In fact, differently from sensors, the actuators have a close relation between the actuation intensity and the resource consumption. Hence, it is crucial to model the impact of a certain command to the actuators on the power and mass budgets. Performance models are used to verify how the GNC system works, highlighting the impact on the resources of the whole spacecraft. They are also useful to discover possible bottlenecks of the GNC design given the performance and the maximum operative limits of the selected actuators. Performance models use equivalent generic mathematical models, and they commonly model only the principal and the most influent limitations of the actuators.

In analogy with sensor models, actuator models' interfaces need to be defined according to the specific type of model in use and to its specific application. In fact, actuator models are exploited in accelerated simulators for design and development, but they are also used in real-time test benches for verification purposes. In the first case, equivalent engineering data interfaces are enough, while in the second application scenario, the model input/output should be as close as possible to the real ones. Notwithstanding, it is

suggested to implement an actuator model having in mind all its possible applications, since it is very common to port the model from the accelerated simulation environment (e.g., MATLAB/Simulink) to the real-time one. Finally, we shall consider that even actuator models shall be validated, meaning that they shall be truly representative of the real actuator behaviors.

Errors modeling

Actuator models for GNC applications are based on mathematical models of the internal dynamics governing the actuation principle, including all the main limitations and errors existing inside the actuator components. They take as input the commands elaborated by the control and actuation functions, and they provide as output the resulting forces or torques. The latter are then set as input for the block simulating the spacecraft dynamics.

The format and the characteristics of the input depend on the typology of actuator model, which also influence the way in which the internal processes are modeled. As said, the most complete functional models receive the input as the real components would do, and they include almost every dynamical phenomenon characterizing the actuator. A generic functional actuator model is shown in Fig. 7.1.

Actuator models are strongly dependent on the specific component, in fact the internal dynamics is specific for each actuator type. For instance, reaction wheels models include the dynamics of the electric motor, with its control unit, friction, vibrations, and other disturbances. Or, correspondingly, thruster models consider fluidic valves operations and the thermogas dynamics of the propellant. Although, in general, latencies, offset errors, misalignments, and actuator saturation are modeled for any actuator. The errors due to the specific dynamical evolution are included in the actuator model by inserting the disturbance effects into the internal dynamics block.

Note that actuator commands come in discrete time as computed from the GNC, then the internal dynamics block converts it in continuous time for the input in the spacecraft dynamics. In fact, the actuator output is a real physical quantity existing in continuous time. Despite this, the command

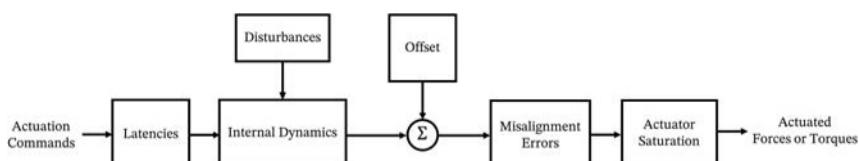


Figure 7.1 Generic functional actuator model.

discretization and the temporal evolution of the internal dynamics may induce a discrete-like actuation output. For instance, the concept of minimum torque or force pulse is related to this effect, and the shortest actuation time is a finite value larger than the minimum possible bit.

Actuator modeling is based on a solid knowledge of the physics behind the actuation principles. Indeed, electric motor dynamics, dynamic and static friction, electromagnetic induction, thermochemical processes, vibrations and rotor balance, fluid dynamics, and so on are just some of the physical effects influencing the internal actuator dynamics. For each specific functional actuator model, we shall understand all the main principles and physical laws governing the actuators.

Performance models do not typically include the internal dynamics, and the ideal forces and torques to be actuated are just processed adding a direct realization of the specific dynamically induced errors. Moreover, the main actuation limitations are included as well, as shown in Fig. 7.2. Performance models are used to quantify the active consumption of resources by the actuators. In fact, these components are commonly the most demanding for the GNC system in terms of resource budgets. Their consumption is strictly related to the actuation intensity, and, thus, performance models include resource consumption models that compute the resource spent for a given commanded force or torque—for instance, the electric power need, the propellant mass flow, or the current intensity. Note that resource consumption models can also be included in the functional actuator models, which are, in this case, referred to as functional-performance models.

Actuator faults

Actuator faults have a huge impact on the GNC functionalities and even on the spacecraft lifetime. In fact, it is more difficult to guarantee many degrees of redundancy on the actuators, than on the sensors. Failed actuators may lead to a partial loss of the controllability or, in the worst cases, to the loss of the entire mission. To minimize these risks, we should include as many

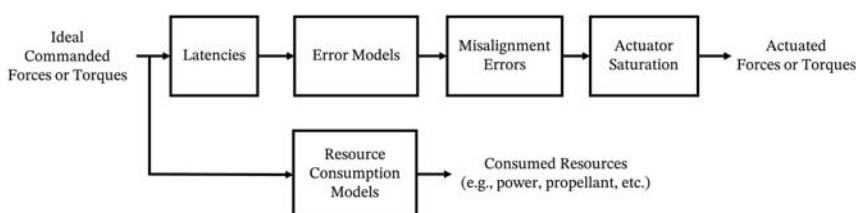


Figure 7.2 Generic performance actuator model.

redundancies as possible, but we shall also include degraded control modes that can exploit the remaining operational components to achieve at least a survival model stabilization. Then, the knowledge of typical actuator faults is fundamental to properly deal with the GNC design in faulty conditions.

A failed actuator is commonly switched off and removed from the operational actuation set, but the faulty units can be difficult to be identified. In fact, the actuator forces and torques are not directly sensed back or measured by the GNC, and, unless the component has an internal error feedback indicator, the failure shall be detected from anomalies in the spacecraft dynamics [6–8].

The actuator faults modeling is strongly dependent from the specific actuator and its internal dynamics, and it is difficult to generalize the fault behaviors as done for the sensors in [Chapter 6 – Sensors](#). Typical actuator faults are commonly related to premature wear on mechanical components, such as bearings, valves, gimbals, or gearboxes. Even electrical elements are susceptible to faults. For example, electric motors or, in general, solenoids (i.e., magnetorquers) may be affected by low resistance fault, due to overheating, physical damage, or degradation of the insulation of the coil windings. Then, insufficient isolation between the windings or conductors may lead to short circuits and, eventually, to complete failure. Similarly, these components can be generally affected by overheating, electrical or mechanical overload, which irreversibly damage the actuator unit. The results of these damages in the actuation output can be categorized in:

- Decreased actuation force or torque.
- Increased bias force or torque.
- Oscillatory and irregular force or torque.
- Continuous generation of actuation force or torque.
- Failure to respond to control signals.

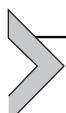
The last two faulty conditions typically determine the loss of the component that shall be switched off and excluded from the actuation control functions. The first three may be dealt with dedicated control techniques.

To make a practical example of actuator faults, momentum exchange devices are considered. In particular, we focus on reaction wheels. They are the most common actuators for spacecraft attitude control, and they are also among the spacecraft components more prone to fail. Indeed, at least one degree of redundancy is always included with these actuators. The most typical reaction wheels faults are:

- Random increase in reaction wheel motor current. This is typically due to some hardware level failure in the motor driver unit, and it determines sudden variations of control torque.

- Increase in friction or vibrations. This is the most common fault due to the wear of bearing material over time or due to some problems in the lubrications of the component. The result is an increased torque bias and torque noise.
- Bus voltage failure at high speed. This fault may take place when the bus voltage is low and a large back electromotive force is present because the reaction wheel motor is operating at a high speed. This fault limits the motor current and, consequently, the motor torque.
- Torque oscillations and sudden steps at low frequency. This is due to variable lubricant dynamics and to bearing nonlinear friction when the reaction wheels are operated at low speed.

Actuator faults need to be simulated in order to reproduce some failure scenarios that resembles real-life faults and failures by utilizing the concept of fault injection into the simulated system. Thus, actuator models shall be capable to reproduce the possible faults that may occur while on orbit with the purpose to design and analyze GNC systems with fault detection and recovery capabilities.



Thrusters

In order to control the spacecraft position, to launch a body in space, or to maneuver from an orbit to another, we need a control force. This control force is generated from the acceleration of mass through the nozzle of a propulsion system, which results in an equal and opposite force — as dictated by Newton's third law. This force is defined as thrust, and it is generated by an actuator denoted as thruster. The propulsion subsystem is needed for position and orbit control purposes, but also for attitude control. Indeed, thrust can be generated with a lever arm with respect to the spacecraft's CoM. There are a few design factors that must be well-accounted for the propulsion subsystem design process, which will heavily depend on the specific mission. Perhaps one of the most pertinent ways to interpret the performance of the propulsion system is in terms of the velocity increment capability, referred to as the Delta-*V* or ΔV . This is computed using Tsiolkovsky's *rocket equation*:

$$\Delta V = I_{sp}g \ln\left(\frac{m_i}{m_f}\right)$$

where $m_f = m_0 - m_p$ is the final spacecraft mass, m_0 is the initial mass, m_p is the mass of propellant on-board, g is Earth's gravitational acceleration, and

I_{sp} is the specific impulse of the thruster. Specific impulse is a key design parameter in thruster design — it is the ratio of the thrust, F , to the weight flow rate of thruster propellant, $\dot{m}g$. This value conveys how efficiently the fuel is turned into thrust by the propulsion system.

$$I_{sp} = \frac{F}{\dot{m}g}$$

where \dot{m} is the mass flow rate.

For preliminary design purposes, an alternate form of this equation is often used, which allows the calculation of the mass of fuel needed for a desired ΔV .

$$m_p = \left[e^{\left(\frac{\Delta V}{I_{sp}g} \right)} - 1 \right] = m_0 \left[1 - e^{\left(-\frac{\Delta V}{I_{sp}g} \right)} \right]$$

The total ΔV mission requirement will incorporate all operational aspects such as: orbital insertion, maneuvers, and corrections, in addition to attitude control. For more complex, safety critical missions, there may also be additional constraints such as Collision Avoidance Maneuvers (CAMs) that must also be allocated for within the ΔV budget. A thruster with high I_{sp} will therefore use less propellant. Other important performance requirements may include propulsion operating pressure, thruster redundancy, and propellant mass. Constraints may be imposed due to the physical limitations of the spacecraft design, with respect to thruster position and orientation. For instance, plume impingement, described in [Chapter 3](#) — The Space Environment, is a critical concern for many missions (especially where optical instruments are involved), so there could be some limitations in terms of thruster configurations, and this may consequently impact propulsion performance.

Thrusters assembly

In broad terms, propulsion is generally divided into three subcategories: cold gas, chemical, or electric propulsion. Alternatives such as solar sails, nuclear, etc., will not be covered here but are also worth mentioning [9–11]. Cold gas propulsion systems are the simplest of all: consisting of a controlled, pressurized gas and a nozzle. Cold gas has a low performance but is favored when simplicity is the priority, in addition to low precision needs. Chemical propulsion usually involves a chemical reaction, and it can include the following

subcategories: liquid, solid, and hybrid, depending on the state of the propellant. In general, when referring to solid systems, the term *motor* is used, whereas liquid or hybrid systems are referred to as *thrusters* or *engines*. Chemical propulsion is the most widely used within the space industry, since different levels of performance can be achieved using a variety of propellants and mixing ratios (e.g., monopropellant, bipropellant, or hybrid) [12]. Finally, electric propulsion works by accelerating the mass to a high speed using electrostatic or electromagnetic fields. Since electric thrusters have a higher I_{sp} , they typically offer greater fuel efficiency (i.e., fuel consumption is lower) than chemical propulsion options, but offer lower thrust capability due to their limited power. A brief overview of these options is given in Table 7.1; however, a more comprehensive review of each propulsion type and their specific advantages and disadvantages is detailed in Ref. [12].

Table 7.1 Overview of the main propulsion types.

Propulsion				
type	Typical I_{sp} (s)	Uses	Strengths	Weaknesses
Cold gas	30–70	Attitude control, orbit maneuvers, and station keeping	Simplicity, reliability, very inexpensive	Low performance, weight of system
Solid	280–300	Orbit insertion	Simplicity, reliability, inexpensive	Nonadjustable performance, safety concerns
Liquid	Monopropellant: 220–240 Bipropellant: 305–310	Orbit insertion (bipropellant only), attitude control, orbit maneuvers, and station keeping	High/very high performance	Complexity, toxic (some propellant combinations)
Hybrid	250–340	Orbit insertion, orbit maneuvers, and station keeping	Safety, throttable nontoxic	Requires oxidizer fuel system, heavier
Electric	300–3000	Orbit insertion (apogee burn only), orbit maneuvers, and station keeping	High performance, low thrust	Complexity, higher development costs

Thrust management and actuation function

As previously discussed, in order to control the spacecraft, a propulsion subsystem is necessary to perform the required actuation. The relationship between the GNC subsystem, actuators, and other subsystems is depicted in Fig. 7.3. The GNC subsystem (comprising Guidance, Navigation, and Control functions) will output the required force and torque commanded by the controller, and this will be used by the Thrust Management Function (TMF), which will compute the corresponding thruster activations and on-times. This information can subsequently be used to command the thrusters on-board the spacecraft. The realized forces and torques will alter the spacecraft dynamics in order to achieve the desired command (in position and/or attitude).

The TMF is a critical component of the GNC subsystem, and its functional scheme is represented in Fig. 7.4. Its purpose is to translate the controller commanded forces and torques into the corresponding commands for each individual thruster. The position controller commands forces, while the attitude controller requests torque – this being the case for all three axes, which results in a 6 Degree of Freedom (DoF) control capability. The output of the function will consist of a series of “on/off” commands, depending on the thruster’s position and orientation in relation to the spacecraft’s CoM at a given moment.

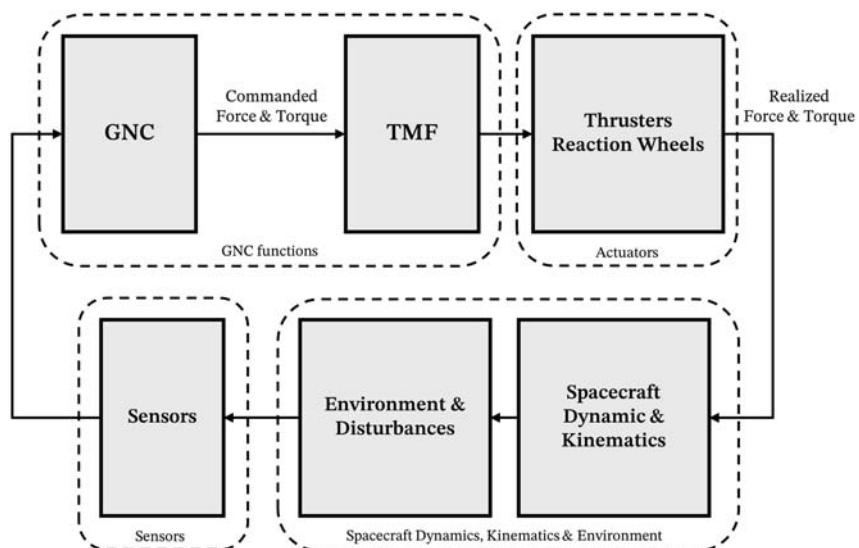


Figure 7.3 Schematic of closed-loop system and relationship between GNC and thrusters.

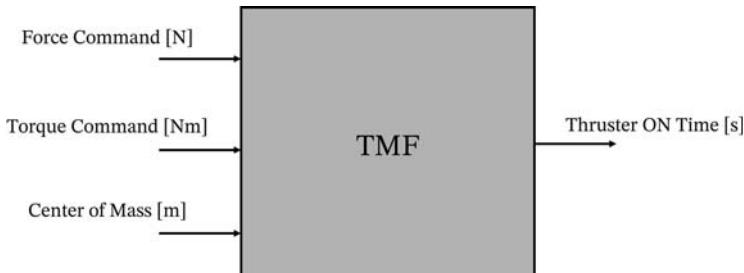


Figure 7.4 Schematic of an example TMF to compute thruster on-times.

The output of the TMF is used directly by the actuators, but there are some important limitations introduced by the thruster hardware, namely [13]:

- Coupling of forces and torques due to the thruster layout (usually due to accommodation and/or redundancy constraints) means it is not always possible to use dedicated thrusters for specific axis control, instead a more complex mapping of thrusters is required.
- On/off nature of certain thrusters means that the thrust output can, in some cases, be either nominal or zero — although some types of thrusters ensure a certain throttle threshold is maintained.

For an ideal thruster configuration, for which there is a clear correlation between each thruster and the control of an individual axes in both force and torque, a “look-up” table may be used. The purpose of this type of table is to provide the mapping between each thruster and the specific control commands. Unfortunately, this method will only be valid for simple thruster configurations, where the CoM is fixed throughout operation. Another drawback of this approach is that it will not be the optimal solution — with respect to fuel consumption nor precision. One benefit of this approach is the speed and relatively small CPU requirements, although this can quickly change when redundancy constraints are applied, as an additional table must be added for all possible failure scenarios, as discussed in Ref. [13].

To illustrate the relationship between thrusters and their corresponding control authority, let us consider a relatively simple thruster configuration example. It consists of eight thrusters that will provide 5 DoF control of the spacecraft.

Fig. 7.5 shows the thruster layout relative to the origin (note this location differs from the CoM location, which is important for subsequent analysis). Each thruster is mounted at an angle of 60 degrees azimuth, ϕ , and 0 degrees

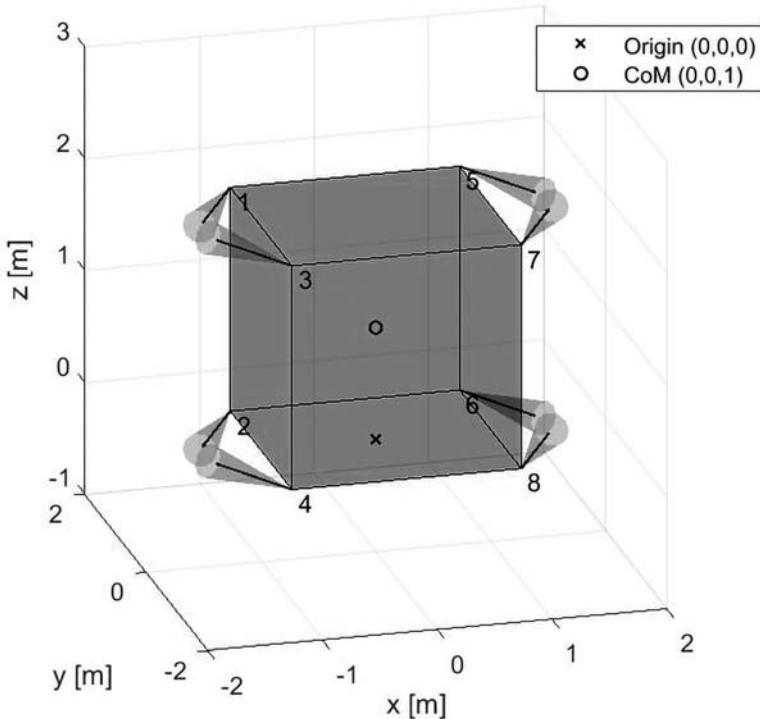


Figure 7.5 An example of a 5 DoF thruster configuration.

elevation, θ ; with varying orientations to cover the required DoFs. The arrow indicates the plume direction of the thruster, where the resulting thrust direction vector is acting in the opposite direction toward the spacecraft body. For the direction vector definition detailed in [Table 7.2](#), note that both angles are defined in radians.

Table 7.2 Example thruster configuration data.

Thruster	Position [m]				Direction [-]		
1	-1	1	2	$\cos(-\theta) \cdot \cos(\phi)$	$\cos(-\theta) \cdot \sin(\phi)$	$\sin(-\theta)$	
2	-1	1	0	$\cos(\theta) \cdot \cos(\phi)$	$\cos(\theta) \cdot \sin(\phi)$	$\sin(\theta)$	
3	-1	-1	2	$\cos(\theta) \cdot \cos(-\phi)$	$\cos(\theta) \cdot \sin(-\phi)$	$\sin(\theta)$	
4	-1	-1	0	$\cos(-\theta) \cdot \cos(-\phi)$	$\cos(-\theta) \cdot \sin(-\phi)$	$\sin(-\theta)$	
5	-1	-1	2	$-\cos(\theta) \cdot \cos(-\phi)$	$-\cos(\theta) \cdot \sin(-\phi)$	$-\sin(\theta)$	
6	1	1	0	$-\cos(-\theta) \cdot \cos(-\phi)$	$-\cos(-\theta) \cdot \sin(-\phi)$	$-\sin(-\theta)$	
7	1	-1	2	$-\cos(-\theta) \cdot \cos(\phi)$	$-\cos(-\theta) \cdot \sin(\phi)$	$-\sin(-\theta)$	
8	1	-1	0	$-\cos(\theta) \cdot \cos(\phi)$	$-\cos(\theta) \cdot \sin(\phi)$	$-\sin(\theta)$	

Table 7.3 Example of simplified thruster activation “look-up” table.

	Thruster activated							
	1	2	3	4	5	6	7	8
F_x	x		x	x				
F_y	x		x		x		x	
F_z								
T_x		x		x		x		x
T_y	x			x		x		
T_z			x		x			x

This configuration can easily be adapted to a look-up table approach due to the fact that the orientation of the thrusters corresponds directly to each axis of control. This is in part due to the zero-elevation angle, but also resulting from the CoM being located equally between the thrusters and also assuming that this location does not change during operation. Referring to Table 7.3, it is possible to see the direct correlation between each thruster and each axis of control. Note that control in the F_z direction is not possible due to the elevation angle of the thrusters being zero.

This relatively simple and direct correlation is not always the case for a number of reasons, some of which have been mentioned and therefore it can be a little more difficult to directly interpret which thrusters are activated for which control action. For this reason, on-board computation is necessary, and this is performed using linear programming techniques, using an algorithm known as the Simplex [14].

The Simplex algorithm allows you to compute the normalized open ratios of each thruster, \mathbf{y} , and from this you may compute the on-time t_{on} (in seconds) for each thruster, using the thruster sample time, t_s . The on-time will be limited by the minimum possible thrust duration, which will depend on the characteristics of the thruster itself; specifically, the Minimum Impulse Bit (MIB). The MIB is the minimum achievable pulse duration at a certain thrust level and therefore dictates the precision of the thrusters:

$$t_{on} = \mathbf{y} \cdot t_s$$

where t_{on} and \mathbf{y} will consist of the same number of elements as the number of thrusters, n .

The thruster allocation problem for n thrusters can therefore be defined as such:

$$\begin{bmatrix} F_{1x} & F_{2x} & F_{3x} & F_{4x} & \cdots & F_{nx} \\ F_{1y} & F_{2y} & F_{3y} & F_{4y} & \cdots & F_{ny} \\ F_{1z} & F_{2z} & F_{3z} & F_{4z} & \cdots & F_{nz} \\ T_{1x} & T_{2x} & T_{3x} & T_{4x} & \cdots & T_{nx} \\ T_{1y} & T_{2y} & T_{3y} & T_{4y} & \cdots & T_{ny} \\ T_{1z} & T_{2z} & T_{3z} & T_{4z} & \cdots & T_{nz} \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \vdots \\ \gamma_n \end{bmatrix} = \alpha \begin{bmatrix} F_x \\ F_y \\ F_z \\ T_x \\ T_y \\ T_z \end{bmatrix}$$

$$\boldsymbol{\tau} = [F_x F_y F_z T_x T_y T_z]^T$$

where the torsor, $\boldsymbol{\tau}$, generated by the thrusters is defined in x, y, z components for the force, F_i and torque, T_i . The normalized open ratios, $\gamma_i \leq 1$, are given for each thruster. A scaling factor, α , is used as part of the optimization criteria, where the cost function, J , is defined as a minimal fuel problem, while maximizing the scaling factor, α .

$$J_{\max} = - (c_1 \gamma_1 + c_2 \gamma_2 + c_3 \gamma_3 + c_n \gamma_n) + c_\alpha \alpha$$

where $c_i (i = 1 : n)$ is a coefficient denoting propellant consumption and c_α is a weighting factor [13].

From this Simplex optimization procedure, the open ratios for a given torsor command, $\boldsymbol{\tau}_{cmd}$, are computed. Once you have the open ratios and on-times, you may subsequently compute the realized output torsor, $\boldsymbol{\tau}_{realized}$. One of the steps in the “mapping” between thrusters and the corresponding torsors is to generate the influence matrix (or mixing matrix), \mathbf{A}_{inf} . This matrix relationship shows us the relationship between each thruster and each DoF.

$$\boldsymbol{\tau}_{realized} = \mathbf{A}_{inf} \cdot \mathbf{y}$$

In order to compute \mathbf{A}_{inf} , which is a $6 \times n$ matrix, we must first define a position vector $\mathbf{r}_i = [x \ y \ z]^T$ and a direction vector $\mathbf{d}_i = [u \ v \ w]^T$ such as those specified in the example given in Table 7.2. We must also define the thruster position taking into account the CoM location, where:

$$\mathbf{p}_i = \mathbf{r}_i - \mathbf{r}_{CoM} = [P_x \ P_y \ P_z]^T$$

$$\mathbf{A}_{inf} = \begin{bmatrix} \frac{u_1}{|\mathbf{d}_1|} & \frac{u_2}{|\mathbf{d}_2|} & \frac{u_3}{|\mathbf{d}_3|} & \dots & \frac{u_n}{|\mathbf{d}_n|} \\ \frac{v_1}{|\mathbf{d}_1|} & \frac{v_2}{|\mathbf{d}_2|} & \frac{v_3}{|\mathbf{d}_3|} & \dots & \frac{v_n}{|\mathbf{d}_n|} \\ \frac{w_1}{|\mathbf{d}_1|} & \frac{w_2}{|\mathbf{d}_2|} & \frac{w_3}{|\mathbf{d}_3|} & \dots & \frac{w_n}{|\mathbf{d}_n|} \\ P_{x1} \times \frac{u_1}{|\mathbf{d}_1|} & P_{x2} \times \frac{u_2}{|\mathbf{d}_2|} & P_{x3} \times \frac{u_3}{|\mathbf{d}_3|} & \dots & P_{xn} \times \frac{u_n}{|\mathbf{d}_n|} \\ P_{y1} \times \frac{v_1}{|\mathbf{d}_1|} & P_{y2} \times \frac{v_2}{|\mathbf{d}_2|} & P_{y3} \times \frac{v_3}{|\mathbf{d}_3|} & \dots & P_{yn} \times \frac{v_n}{|\mathbf{d}_n|} \\ P_{z1} \times \frac{w_1}{|\mathbf{d}_1|} & P_{z2} \times \frac{w_2}{|\mathbf{d}_2|} & P_{z3} \times \frac{w_3}{|\mathbf{d}_3|} & \dots & P_{zn} \times \frac{w_n}{|\mathbf{d}_n|} \end{bmatrix}$$

The error in the realized torsor, $\boldsymbol{\tau}_{error}$, is then simply:

$$\boldsymbol{\tau}_{error} = \boldsymbol{\tau}_{cmd} - \mathbf{A}_{inf} \cdot \mathbf{y} = [F_{x_{error}} F_{y_{error}} F_{z_{error}} T_{x_{error}} T_{y_{error}} T_{z_{error}}]^T$$

where $\boldsymbol{\tau}_{error}$ is a column vector of dimension 6×1 .

Use of the Simplex algorithm on-board requires precise knowledge of the CoM at all points in flight. If this is inaccurate, large errors will be evident between the commanded and realized torsors. Errors in the realized torsor will result in trajectory deviations and, most likely, unnecessary fuel consumption. The error can be due to many reasons, some of which are physical issues, such as: bias misalignments incurred during the mounting of the thrusters, discrepancies in the maximum force output by each thruster, or the specified I_{sp} due to manufacturing error margins. Errors in the thruster MIB may also have a significant impact on performance, as the MIB corresponds to the minimum torque that can be commanded to the spacecraft. Uncertainties in the spacecraft mass and inertia can also negatively impact the thrust vector realization – especially if the CoM is inaccurate. The thrust duration (specified by t_{on}) is also an important source of error as there are usually delays in the thrust realization time, which should always be incorporated into the GNC simulations to assess the impact on performance. Thruster timing issues may also be a result of thruster failures, such as when they are stuck on or off – for instance, when a valve becomes blocked in place. This is a critical issue that must be dealt with swiftly and normally with help of the FDIR functions on-board. If the thrusters cannot be reconfigured after a failure and mission safety is threatened, a CAM may be executed. Additional complexities occur when effects such as fuel sloshing

Table 7.4 Thrusters strengths and weaknesses.

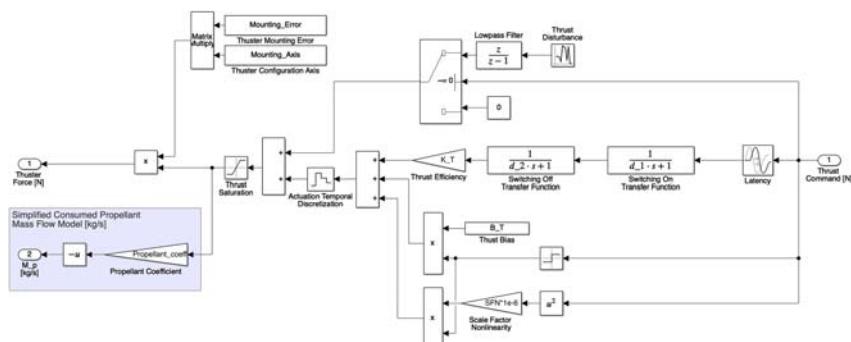
Strengths	Weaknesses
Possibility to generate forces	Complex equipment with fluidics and fuel tanks
Large torques available	Large power demand
Net external torque	Fuel required (e.g., mass, sloshing, center of mass variation)
Three axis-control possible	Forces and torques poorly (or non-) throttleable

and flexible modes impact the dynamic behavior of the spacecraft. Again, these can alter physical properties such as the CoM, which will in turn lead to thrust vector errors. Proper thruster performance is only guaranteed if all these uncertainties are well understood and incorporated into the both the propulsion and GNC subsystem design process. This involves accurate modeling of all errors within the closed-loop system, so their impact can be fully understood, but also to enable design of a robust controller that is able to compensate for some of these effects. Evidently, there will always remain an element of uncertainty, but the task is to minimize this wherever possible.

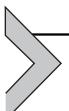
Table 7.4 summarizes the strengths and weaknesses of thrusters.

Thrusters model

A detailed functional thruster model shall consider the specific thruster typology. In the case of a simplified thruster model, the main effects can be modeled as in the example reported in Fig. 7.6. The input command is directly the desired thrust value, which is then processed to include the latency and the switching dynamics induced by the fluidics and the ignition

**Figure 7.6** Example thruster model.

system. In fact, two low-pass filters are present to model the noninstantaneous variations of the thrust, both in the switching on and in the switching off dynamics of the thruster. Then, the thrust is scaled according to the thrust efficiency factor, it is limited by a saturation range, and it is modulated with a temporal discretization. The latter reproduces the digital nature of the thruster control electronics, and it helps in modeling the minimum thrusting interval. Moreover, the thrust is biased in a way that the minimum thrust value is greater than zero when the thruster is active. Similarly, the force has a nonlinear scale factor error proportional to the command magnitude. When a thrust is commanded, the output is also disturbed with a pink noise random error. Note that when the thruster is not active, the output force is zero. Finally, the thrust magnitude is converted in a force vector considering the mounting axis, with mounting errors. The propellant consumption model is proportional to the actual thrusting force.



Reaction wheels

Reaction wheels are the most common and versatile actuator to control the spacecraft rotational state. They are used almost on any orbit and satellite class for precise attitude control, both for nominal modes and in some architectures, for safe modes. Their working principle is based on the internal angular momentum exchange between the satellite main body and the wheels themselves. The overall angular momentum of the spacecraft is:

$$\mathbf{h}^b = \mathbf{I}\boldsymbol{\omega}^b + \mathbf{h}_{\text{wheels}}^b,$$

where \mathbf{I} is the moment of inertia tensor of the spacecraft including the inertia of the wheels. Then, $\mathbf{h}_{\text{wheels}}^b$ represents the relative angular momentum of the wheels along their spin axes with respect to the spacecraft body. If no external torque is applied, the conservation of angular momentum reads as:

$$\dot{\mathbf{h}}^b = 0 \rightarrow \mathbf{I}\dot{\boldsymbol{\omega}}^b = -\dot{\mathbf{h}}_{\text{wheels}}^b. \quad (7.3)$$

In the previous equation, without loss of generality, the inertia of the spacecraft has been assumed to be constant, and the spacecraft to be in simple rotation around one of its principal inertia axes.

The change in the wheel's angular momentum is due to the torque directly applied on the wheel by the driving electric motor and by the disturbing loss effects:

$$\dot{h}_{\text{wheel}} = I_{\text{wheel}} \dot{\omega}_{\text{wheel}} = t_{\text{wheel}} = t_{\text{motor}} - t_{\text{loss}},$$

where I_{wheel} is the flywheel inertia, $\dot{\omega}_{wheel}$ is the change of relative wheel's speed with respect to the spacecraft body, and t is a torque about the spin axis of the wheel. The subscript "motor" indicates the motor torque, and "loss" indicates the loss torque (i.e., the sum of the friction torque and all the other losses, which will be detailed in the followings). Note that the torque applied on the wheel enters the spacecraft dynamics with the negative sign, as in Eq. (7.3). The reaction wheels exploit Newton's third law of motion of action-reaction. In fact, t_{wheel} is an *internal* torque and the reaction wheels cannot exchange a net angular momentum externally with the environment. In addition to the reaction torque, the angular momentum stored in the flywheels, \mathbf{h}_{wheels}^b , enters in the attitude dynamics being an additional contribution in the angular momentum of the spacecraft. Reaction wheels can be used also as momentum wheels: this indicates a way of using wheels not to exchange torque but as momentum storage. This means that the wheel is kept at constant speed, as in dual spin satellites.

Reaction wheels assembly

The reaction wheels are physically composed by a flywheel mounted on a bearing assembly and put in motion by an electrical motor. The motor is driven by the wheel electronics which interfaces with the satellite for telemetry, telecommand, and power. The bearing is composed by a shaft, ball bearings, and lubricant for durability and friction reduction. Reaction wheels with hybrid bearings without lubricant are becoming more popular in modern applications. Moreover, there have been studies on magnetic levitation bearings, but operational applications seem still unpractical.

In terms of configuration, the flywheel assembly — which is composed by the flywheel, the bearing, the motor, and a mechanical case — can either be mounted on top of the wheel drive electronics — which is composed by the power management and the electronics to process command, drive the motor, and generate telemetries — or separately. Usually, smaller wheels come in an integrated assembly, while on larger wheels, a separated electronic can be more convenient to optimize accommodation on the satellite. In these regards, the flywheel assembly introduces on the satellite a mechanical disturbance, as discussed in the internal perturbation section of Chapter 6 — Sensors, while the wheel drive electronics is one the most thermally dissipative units. This is especially true during wheel braking due to the conversion of mechanical energy in thermal energy. Some reaction wheel models are sealed to maintain a vacuum in the flywheel assembly in order to avoid any possible

contamination of the bearings and higher friction levels when the wheel is operated on ground for testing purposes.

Knowledge of the flywheel speed is needed to properly drive the wheel. The most common technologies involve rotary encoders based on:

- Optical detectors, reading markers on the flywheel.
- Hall effect sensors, where permanent magnets are placed opportunely on the flywheel to generate a pulse when crossing a motor pole.

Optical detectors and Hall effect sensors generate a pulse train that is used for the speed estimation, although this is a complex task since there are several error sources:

- During the estimation period, the wheel speed can vary due to the application of a torque.
- The optical markers and magnets are not perfectly equally spaced on the flywheel, thus introducing a mechanical jitter in the signal reading even at constant speed. When the wheel performs multiple rotations during a sampling period, it is possible to use the same marker to measure the rotation period and remove the contribution of the mechanical jitter. At low speeds, this is not possible.
- Electrical jitter can be introduced by the optical detector or Hall effect sensor's acquisition chains.

The speed estimation can be performed by the on-board computer, which receives the pulse train, or directly by the wheel electronics. The first option is common in wheels with analogic interfaces, while the second in wheel with a digital interface.

Reaction wheels electronics could be equipped with an internal control loop that regulates the wheel speed and torque compensating for errors introduced by the motor or variations in the friction. This is commonly done in digital components. Otherwise, this functionality can be implemented directly in the GNC algorithms for analogic actuators, which are commonly just commanded in desired torque. The most common motor control logics are based on a cascade of proportional integral (PI) controllers. They regulate motor voltage and current to achieve the desired speed and torque. This method improves the actuator performance minimizing the deviations between desired and actual reaction wheel torque.

From an operational point of view, there are several aspects to be considered. When the reaction wheel is not spun for a long period on ground, the lubricant accumulates inside the bearing because of the gravity. Since the lubricant is not uniformly distributed, there is an increase of friction due to direct metal to metal contact between the ball bearing and the cage.

Operations of the wheel at high speed for several hours allow to recover the nominal behavior and performance of the wheel. This operation is called run-in, and it is needed before any test on ground where the nominal performance of the wheel is needed. Manufacturers provide recommendations on how to perform the run-in.

Run-in is typically needed after release from the launcher since the lubricant is usually not uniformly distributed after launch and the friction torque is very high. Lubricants introduce constraints on the minimal operative temperature. After long periods of inactivity in orbit, the wheel bearing could reach temperatures below this threshold. It is therefore necessary to heat the wheel, either using the satellite thermal control or switching on and keeping in standby the wheel itself. Reaction wheels with hybrid bearings without lubricant, despite higher friction, are less complex and do not require these dedicated operations.

Wheels are life limited in terms of number of revolutions (e.g., typical value: 100–1000 million revolutions) and zero-crossing events (e.g., typical value: 10–100 thousand). A zero-crossing is defined as a change of rotation direction (i.e., from clockwise to anticlockwise or vice versa), since for a short instant of time, the wheel has zero speed. Under this condition, the friction in the wheel increases and the wheel experiences a small mechanical shock, which could create an audible sound on ground and disturb the satellite attitude in orbit. There are several strategies to limit the occurrence or effect of zero-crossing (e.g., null-space controller) or to limit their effect (e.g., cross-traveler controller). In general, it is a good practice to operate the wheel imposing a bias in the nominal spin rate. Indeed, operations of reaction wheels at low speed shall be avoided. At low-speed reaction, wheel performance is worse, both for friction instabilities and for the difficulty to estimate the wheel speed and, therefore, the momentum.

Friction and microvibrations

Friction has a major impact on wheel performance and power consumption. The complete description of internal friction is a complex task, but the main characteristics can be outlined with a brief introduction to the tribology. Friction in reaction wheel bearings is described by four main terms:

- Coulomb friction is a constant opposing force independent of velocity.
- Viscous friction is proportional to velocity, and it is zero when velocity is null.
- Stiction is defined as the torque required to start a motion from rest, and it is greater than the Coulomb dynamic friction.

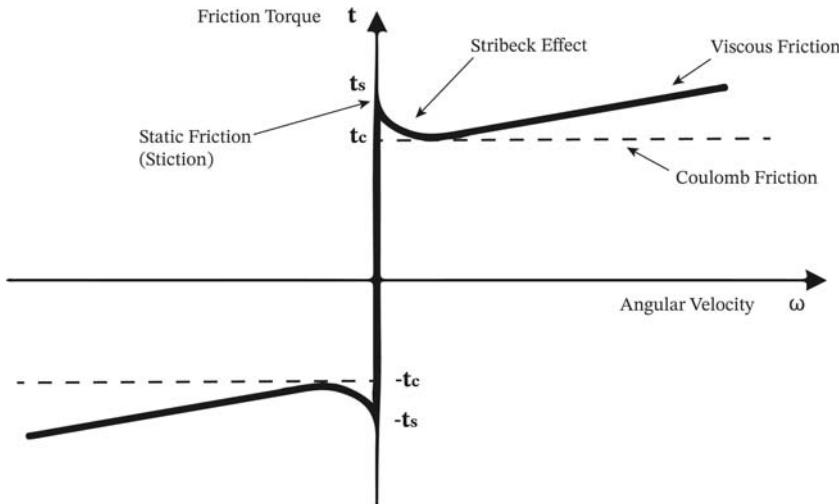


Figure 7.7 General kinetic friction model.

- Stribeck effect is associated to the continuous passage from static to dynamic friction. It describes the decreasing of friction for an increasing velocity at low spin rates. It is generally modeled with an exponential function.

A combination of stiction, Stribeck, Coulomb, and viscous friction terms is commonly referred to as General Kinetic Friction (GKF) [15]. The GKF model may approximate the real friction torque with a degree of confidentiality of 90% [16], and it is shown in Fig. 7.7. Note that the computed friction term enters the wheel dynamics in the loss term with a negative sign. Moreover, to avoid discontinuities in the force at zero spin rate, the negative and positive stiction points can be connected with a steep straight line with finite slope, as proposed by Karnopp [17].

Friction instabilities can occur at any speed regime and are a performance-limiting factor for missions with high pointing requirements. There are three main types of instabilities:

- Friction jumps. This can be seen as a form of friction coefficient instability, usually happening at high-speed regimes, due to mechanical instability of the bearing. The friction can change by a few mNm for a period of several hours. From an attitude point of view, this is equivalent to a step disturbance force.
- Oil jogs. This is a sudden change of friction in the shape of a spike due to accumulation of lubricant between the ball bearing and the cage.

The friction can change by a few mNm for a period of few seconds. From an attitude point of view, this is equivalent to an impulse disturbance force.

- Torque noise. This is the superimposed effect of several contributors (e.g., bearing imperfections, ripple noise or other forms of electrical noise in the motor, electromagnetic compatibility disturbances, etc.). Given its high-frequency contribution, it has usually no major effect on the satellite dynamics, although it injects noise in the wheel speed estimation process and therefore in the control loop. Adequate low-pass filtering technics shall be implemented.

As any mechanism on a satellite, reaction wheels are a source of mechanical disturbance and vibrations. Therefore, they are usually placed away from sensitive elements like payloads. There are three main types of disturbance, often referred with the collective term of *microvibrations*.

- Static imbalance acts as an in-plane centrifugal force due to the instantaneous rotation axis not passing through the flywheel CoM. This force is proportional to the square of the wheel speed. The application of this force results in a torque also proportional to the distance from the wheel to the CoM of the spacecraft. In SI units, static imbalance is measured in kg m, even if it is usually given in g cm [18,19].
- Dynamic imbalance acts as a disturbance torque due to misalignment between the instantaneous rotation axis and the flywheel principal axis of inertia. This torque is proportional to the square of the wheel speed. It is a radial torque, constant in the wheel frame, but rotating with angular velocity ω_{wheel} in the spacecraft frame. In SI units, dynamic imbalance is measured in kg m², even if it is usually expressed in g cm² [18,19].
- Other sources of high frequency disturbance, such as motor noise, bearing disturbances, friction instabilities, etc., are sometimes referred directly as microvibrations. Although these contributors are of small magnitude, there are amplification effects due to the wheel resonance modes. There are three main modes:
 - axial translational mode of the flywheel.
 - parallel rocking mode, rocking the flywheel parallelly to the shaft.
 - antiparallel rocking mode, rocking the flywheel opposite to the shaft.

These perturbing terms directly enter the spacecraft dynamics, as generated by the reaction wheels, and do not directly affect their dynamics. Generally, their impact might be reduced with a proper accommodation on-board and with dedicated control techniques.

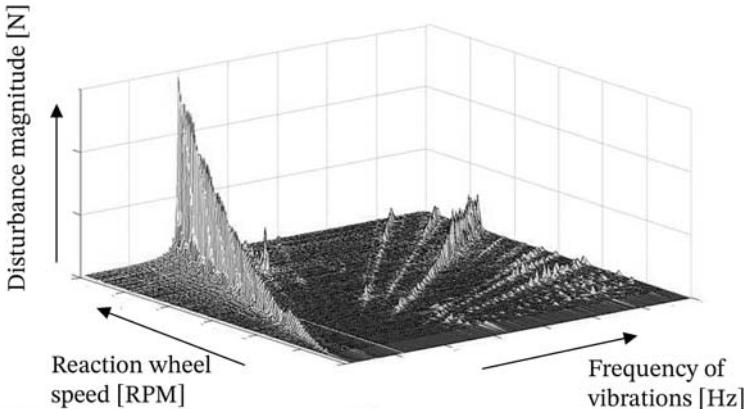


Figure 7.8 Reaction wheels waterfall plot.

The reaction wheel microvibration spectrum can be visualized in waterfall plots, showing the magnitude of the disturbance at different frequency for different rotating speeds, as in Fig. 7.8. Static and dynamic imbalances are partially corrected by the manufacturer balancing the wheel. This is performed by correcting the inertia properties of the flywheel (e.g., by screwing in appropriate locations additional small masses). Note that it is not possible to completely remove the imbalance, since a lot of phenomena play a role: change of properties and shape after sealing of the wheel, deformations due to temperature changes, centrifugal forces at the different rotation speed, mechanical testing, transportation, launch, etc. Further details on these points are also described in Chapter 3 – The Space Environment.

Multiple reaction wheels actuation function

To have three-axis controllability of the satellite, it is needed to have at least three wheels active, usually forming a tern aligned as the satellite reference frame. Since wheels are prone to failures, it is common to have four or more wheels on the satellite, either in cold or hot redundancy. The most efficient configuration is when the wheels are placed in a pyramidal configuration. When the wheels are used in hot redundancy, there is an additional DoF that can be used to minimize the power consumption, limit occurrence of the zero-cross, thanks to the null-space theory, or reduce microvibrations with lower nominal spin rates. This is a desired feature for high pointing performance satellites. Therefore, there are satellites with five (or more) wheels in order to preserve this capability even in presence of a failure.

Multiple reaction wheels assemblies are defined according to the configuration matrix, \mathbf{R} , introduced in [Chapter 5 – Attitude Dynamics](#):

$$\mathbf{h}_{\text{wheels}}^b = \mathbf{R} \mathbf{h}_{\text{wheels}}^r,$$

where \mathbf{R} is a $3 \times n$ matrix, whose columns represent the direction of the axis of rotation of the internal rotors in the body frame, and $\mathbf{h}_{\text{wheels}}^r$ is a column vector with n elements representing the angular momentum of each of the n rotors around its spin axis:

$$\mathbf{h}_{\text{wheels}}^r = [h_{\text{wheel}_1} h_{\text{wheel}_2} \cdots h_{\text{wheel}_n}]^T = [I_{\text{wheel}_1} \omega_{\text{wheel}_1} I_{\text{wheel}_2} \omega_{\text{wheel}_2} \cdots I_{\text{wheel}_n} \omega_{\text{wheel}_n}]^T$$

The configuration matrix \mathbf{R} is composed by the column unit vectors expressing the spin axes' directions, \mathbf{r}_{w_i} , in the spacecraft body frame:

$$\mathbf{R} = [\mathbf{r}_{w_1} \quad \mathbf{r}_{w_2} \quad \mathbf{r}_{w_3} \quad \cdots \quad \mathbf{r}_{w_n}].$$

It is reminded that $\mathbf{h}_{\text{wheels}}^r$ and $\mathbf{h}_{\text{wheels}}^b$ may not have the same magnitude because \mathbf{R} is generally not an orthogonal matrix.

In order to compute the multiple wheels actuation, we need to invert the configuration matrix to transform a desired body frame angular momentum or torque vector into the wheel frame. In this way, we can command each reaction wheel to achieve the desired torque and angular momentum (i.e., spin rate). This is very easy in the case there are only three reaction wheels. Indeed, \mathbf{R} would be a 3×3 invertible matrix and the specific actuation commands could be computed as:

$$\mathbf{h}_{\text{wheels}}^r = \mathbf{R}^{-1} \mathbf{h}_{\text{cmd}}^b$$

$$\mathbf{t}_{\text{wheels}}^r = \mathbf{R}^{-1} \mathbf{t}_{\text{cmd}}^b,$$

where $\mathbf{h}_{\text{cmd}}^b$ and $\mathbf{t}_{\text{cmd}}^b$ are, respectively, the desired angular momentum and torque commanded by the GNC in the body frame. Note that \mathbf{R} is invertible because the three wheels shall not lie in a single plane to guarantee the three-axis controllability. Hence, the spin axes' directions form a configuration matrix with full rank.

In the practical case with $n > 3$, there is no unique way to distribute the angular momentum and torque among the different reaction wheels. The most common solutions for the actuation function are the pseudoinverse law and the minimax method [20]. The former is by far the simplest and most used distribution method, and it uses the pseudoinverse, or Moore–Penrose generalized inverse, of \mathbf{R} to compute the actuation commands as:

$$\mathbf{h}_{\text{wheels}}^r = \mathbf{R}^\dagger \mathbf{h}_{\text{cmd}}^b$$

with

$$\mathbf{R}^\dagger = \mathbf{R}^T (\mathbf{R} \mathbf{R}^T)^{-1}.$$

The pseudoinverse method for distributing torque or angular momentum among redundant reaction wheels minimizes the Euclidean norm of the torque or momentum vector in the wheel frame. Namely, it minimizes the sum of the squares of the individual wheel torques or angular momenta. This condition is optimal under an energy perspective, but it does not allow to use the full capability of the reaction wheel array. In the case a different actuation goal is sought, an alternative distribution law shall be used. In general, the pseudoinverse law is relatively simple to be implemented, and it is a good compromise for most GNC applications.

The configuration matrix depends on the assembly configuration of the reaction wheels, which is strongly related to the specific mission requirements. However, most of the spacecraft have four reaction wheels configured in a pyramidal layout or in a three-axes-and-diagonal layout. In the first case, the spin axes of the reaction wheels are oriented along the four side edges of a square pyramid, while in the latter, three rotors are aligned with the principal axes of the spacecraft and the fourth one has equal components along the three axes. In any of the two cases, a failure does not compromise the three-axis controllability.

The configuration matrix of a pyramidal assembly aligned with the x and y axes, inclined of 30 degrees with respect to the xy plane is:

$$\mathbf{R}_p = \begin{bmatrix} -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} & 0 & 0 \\ 0 & 0 & -\frac{\sqrt{3}}{2} & \frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

while, in the case of the three-axes-and-diagonal layout, the configuration matrix takes the form of:

$$\mathbf{R}_t = \begin{bmatrix} 1 & 0 & 0 & \frac{1}{\sqrt{3}} \\ 0 & 1 & 0 & \frac{1}{\sqrt{3}} \\ 0 & 0 & 1 & \frac{1}{\sqrt{3}} \end{bmatrix}.$$

Note that in the case a reaction wheel fails and the GNC shall exclude it from the actuation function, the configuration matrix shall be modified removing the column associated with the failed component.

Reaction wheels performance

Reaction wheels are characterized by two key parameters: maximum torque and stored momentum capacity. The maximum torque indicated by the supplier is usually the motor torque. The reaction torque can be computed based on the friction torque, and it depends on the wheel speed and torque direction (e.g., while breaking, the reaction torque takes the same sign of the friction torque). The combination of the maximum reaction torque of the wheels projected in a certain direction is the *control authority* around that axis, and it contributes to the satellite *agility* (i.e., the capability to perform slews in a certain time). It is measured in mNm and can vary a lot with the size of the wheel, from 0.1 mNm up to 1 Nm, as order of magnitude. The upper limit comes from engineering constraints, since to generate high torques, there are high power consumptions and dissipations. If higher torques are needed, it is more convenient to use the Control Momentum Gyros, described in the next section. The momentum capacity indicates the maximum angular momentum that can be stored in the reaction wheels. It is measured in Nms and can vary a lot with the size of the wheel, from 0.1 mNms up to 200 Nm, as order of magnitude. From the reaction wheel capacity, it is possible to compute for how long a certain torque can be sustained before reaching the maximum speed, which is called *saturation*, since the wheel cannot provide anymore torque in that direction. Maximum rotational speeds are typically in the range from 1000 to 6000 RPM.

It shall be remarked that operating the wheel at high nominal angular rates makes the wheel more power consuming for the same torque level, the microvibrations are higher, and the actuator is more easily saturated. Similarly, low speeds are not suggested for zero-crossing events and friction irregularities. Hence, reaction wheels shall be nominally operated in the middle of their speed range.

The angular momentum stored in the reaction wheels evolves in time because of external disturbances and control dynamics. Secular disturbances, as well as not symmetric control actions, may lead to momentum accumulation or momentum *build-up*. To avoid saturation of the reaction wheels, these need to be periodically offloaded, or desaturated, by actuators that can exchange momentum externally (i.e., magnetorquers or thrusters). These secondary actuators provide a net torque that is capable to counteract

Table 7.5 Reaction wheels strengths and weaknesses.

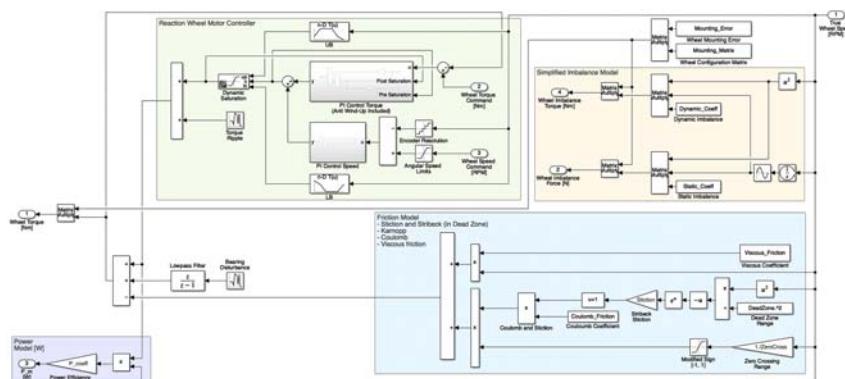
Strengths	Weaknesses
Precise torque delivery	Complex equipment with moving parts and relatively low reliability
Three axis-control possible	Need to be desaturated since it cannot generate external torques
Several architectures available	Several operational constraints (i.e., zero-crossing, low speed, maximum speed, temperature, run-in)
Can be properly oriented to optimize the control authority	Limited maximum torque and momentum capacity due to mass and power constraints Source of mechanical disturbance (i.e., microvibrations)

the reaction torque due to the momentum unloading (i.e., wheel braking). Hence, secondary actuators are mandatory to be used together with reaction wheels, or momentum exchange actuators, because they allow to recover from saturation in order to operate the wheels for indefinite times.

Table 7.5 summarizes the strengths and weaknesses of reaction wheels.

Reaction wheels model

A reaction wheel with digital motor controller is commanded in terms of desired torque and speed. Then, the dedicated motor control unit regulates voltage and current to achieve the commanded values, with feedback on the output torque and speed. An example of this reaction wheel model in MATLAB/Simulink is shown in Fig. 7.9. Specifically, this model contains three main subblocks: reaction wheel motor controller, imbalance model, and friction model.

**Figure 7.9** Reaction wheel model.

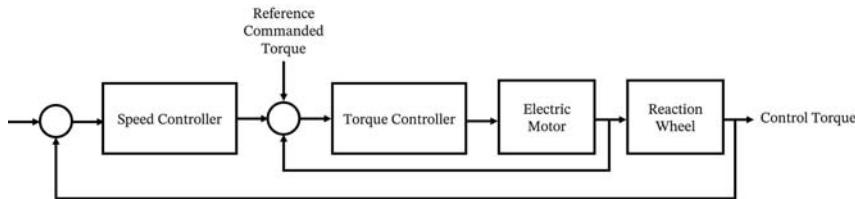


Figure 7.10 Reaction wheel motor controller.

The reaction wheel motor is modeled as a brushless direct current motor, which is very common for reaction wheels. The motor control logic is a classical cascade of PI controllers. As shown in Fig. 7.10, the motor control unit has an inner torque (current) loop and an outer speed loop. The two PI controllers are implemented in each loop to provide fast response and zero steady-state error with respect to the input commands: reference speed and torque. A controlled reaction wheel is capable to guarantee the desired control torque by compensating friction. Hence, it can greatly improve the spacecraft control performance. It shall be noted that the motor controller has a larger control bandwidth on torque, rather than on speed. As a result, the torque is the primary control set-point with a faster settling to the desired torque level. The output is limited by the torque dynamic saturation box that is defined in terms of torque-speed range.

Static and dynamic imbalances are modeled, and they inject perturbation forces and torques directly in the spacecraft dynamics. Thus, the reaction wheel dynamics is not directly dependent on these terms. Although, the control torque is affected by random variations due to the torque ripple and to the bearing disturbances. The friction model is based on the GKF model, including Coulomb and viscous terms. The zero-velocity crossing is characterized by stiction and the Karnopp model, with Stribeck effect in the dead zone. The power consumption model is proportional to motor torque and speed.

The reaction wheels dynamics is represented as:

$$\dot{\mathbf{h}}_{wheels}^b = \mathbf{R}\dot{\mathbf{h}}_{wheels}^r = \mathbf{R}(\mathbf{t}_{motor}(t) + \mathbf{t}_{bearing}(t) - \mathbf{t}_{friction}(\boldsymbol{\omega}_{wheels})),$$

where \mathbf{R} is the wheel configuration matrix, including mounting errors, with respect to the spacecraft body frame. The reaction wheel dynamics is then coupled with the spacecraft dynamics, as discussed in Chapter 5 – Attitude Dynamics.

Control moment gyros

CMGs belong to the class of actuators for attitude control based on the exchange of angular momentum with the spacecraft. Different from reaction wheels, the angular momentum exchange is not achieved by varying the spin rate of the actuator, but by changing the orientation of the spinning axis with respect to the spacecraft body axes. Thus, CMGs generate a torque by gyroscopic effect.

A control moment gyro is constituted by a rotating mass spinning at a constant rate about a gimballed spin axis. The CMG actuation torque is generated by commanding a gimbal rotation around one axis orthogonal to the spin axis of the rotor, and it is directed along the axis orthogonal to both gimbal and spin axes, as in Fig. 7.11. Note that the spacecraft experience the reaction torque, which is opposite to this. The magnitude of the CMG torque, t_{CMG} , depends upon the spin rate of the rotor and the rotation rate of the gimbal axis as:

$$t_{CMG} = I_{CMG}\omega_{CMG}|\dot{\mathbf{s}}| = I_{CMG}\omega_{CMG}\dot{\theta}_G|\mathbf{g} \times \mathbf{s}| = I_{CMG}\omega_{CMG}\dot{\theta}_G,$$

where I_{CMG} is the moment of inertia of the control moment gyro's rotor about the spin axis, \mathbf{s} . ω_{CMG} is the spin rate of the CMG and $\dot{\theta}_G$ is the gimbal rotation rate, around the gimbal axis, \mathbf{g} .

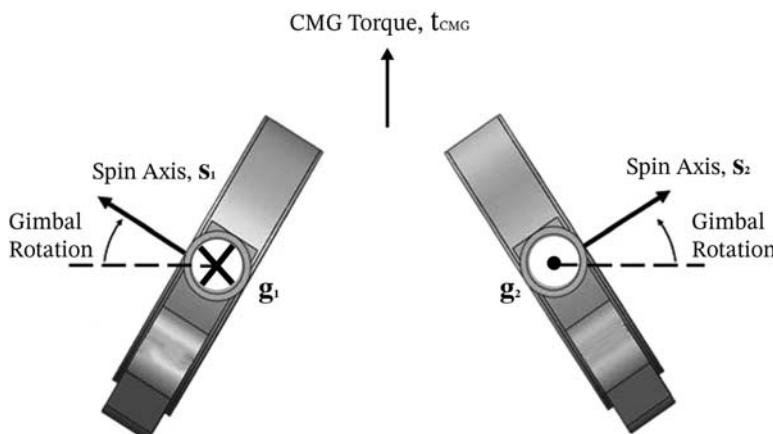


Figure 7.11 CMG scheme for two GMGs with opposite gimbal axes.

At least three single-gimbal CMGs are needed to provide three-axis control of a spacecraft, and the combined effect of a system of n single-gimbal CMG can be expressed as:

$$\mathbf{t}_{CMG} = \sum_{j=1}^n \left(I_{CMG_j} \omega_{CMG_j} \dot{\theta}_G \mathbf{g}_j \times \mathbf{s}_j \right) = \mathbf{C} \dot{\theta}_G,$$

where \mathbf{C} is the CMG angular momentum configuration $3 \times n$ matrix, and $\dot{\theta}_G$ is a column vector with n elements, containing the gimbal rotation rates of each CMG. The configuration matrix \mathbf{C} is time variable since it depends on the time evolution of the spin axes directions around the fixed gimbal axes. If all the CMGs have the same moment of inertia and spin rate, the configuration matrix \mathbf{C} can be expressed as $\mathbf{C} = I_{CMG} \omega_{CMG} \mathbf{C}_G$, where the matrix \mathbf{C}_G is only dependent from the gimbal and spin axes orientation. Assuming four CMGs in a pyramidal configuration as in Fig. 7.12, \mathbf{C}_G is expressed in the body frame as:

$$\mathbf{C}_G = \begin{bmatrix} -\cos\beta \cos\theta_{G_1} & \sin\theta_{G_2} & \cos\beta \cos\theta_{G_3} & -\sin\theta_{G_4} \\ -\sin\theta_{G_1} & -\cos\beta \sin\theta_{G_2} & \sin\theta_{G_3} & \cos\beta \cos\theta_{G_4} \\ \sin\beta \cos\theta_{G_1} & \sin\beta \sin\theta_{G_2} & \sin\beta \cos\theta_{G_3} & \sin\beta \sin\theta_{G_4} \end{bmatrix},$$

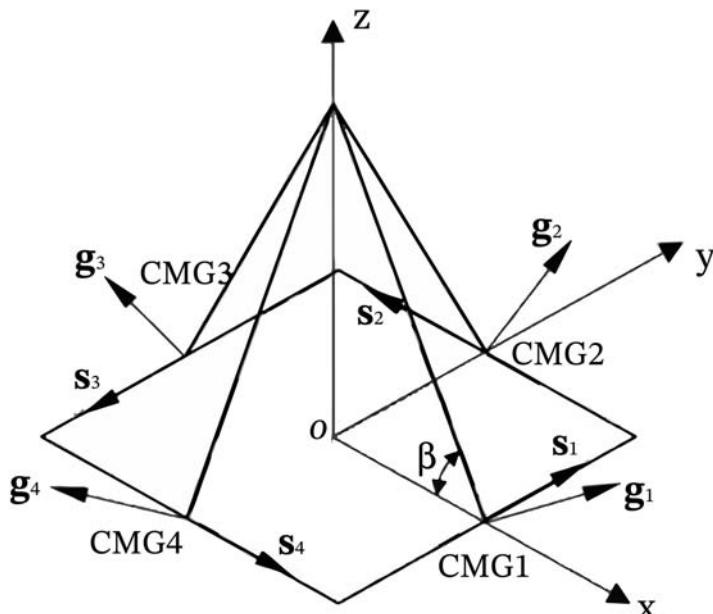


Figure 7.12 Pyramidal 4 CMGs configuration.

where the gimbal angles θ_{G_j} are zero when the spin axes lie on the plane containing the base of the pyramid.

The torque generated by CMGs is generally higher than those obtained with reaction wheels of comparable dimension, and the CMGs are also more efficient under an energy perspective to produce large torques. However, the variation of the actuator configuration due to the rotating gimbal axis leads to a more complex actuation law and some singularity problems arise for specific CMG configuration conditions. When a singular configuration occurs, all the available torques lie in a plane, resulting in a loss of control authority about the axis perpendicular to the singular plane. Several methods to avoid singular configurations have been studied [20–24]. In general, four or more CMGs are needed to avoid passing through the singular points.

CMGs with variable spin rate of the rotor, combining the features of reaction wheels and CMGs, have been investigated, but they are not commonly employed [25,26]. Similarly, CMGs with two gimbal axes have been developed, but they are mechanically complex and the version with a single gimbal axis is by far the most used [26]. Current technology of CMGs allows maximum gimbal rates in the range 0.2–2 rad/s, resulting in maximum allowable torques in the range 0.1–1000 Nm. As a rule of thumb, CMGs are more power efficient than reaction wheels, when the moments of inertia of the spacecraft to be controlled are larger than 10^{-1} kgm². Generally, when the required torque is larger than 0.1–0.5 Nm, a CMG shall be used. Table 7.6 summarizes strengths and weaknesses of CMGs [27].

Table 7.6 CMG strengths and weaknesses.

Strengths	Weaknesses
Large maximum allowable torques	Existence of singular configurations
Large specific torque per stored angular momentum	Mechanically complex
Power efficient for large spacecraft	Large volume per stored angular momentum
Three axis-control possible	Complex actuation laws Need to be desaturated since it cannot generate external torques Source of mechanical disturbance (i.e., microvibrations)



Magnetorquers

Magnetorquers, or magnetic torquers, working principle is based on the electromagnetic interaction with a surrounding magnetic field. These actuators generate a magnetic dipole moment, \mathbf{m} , which is capable to produce a control torque as:

$$\mathbf{t}_{MAG} = \mathbf{m} \times \mathbf{B}, \quad (7.4)$$

where \mathbf{B} is the magnetic field vector interfaced with the magnetorquers. The magnitude of the magnetic torque is thus strongly dependent from the intensity of the external magnetic field, which is a typical limit for the range of application of these actuators. As a matter of fact, magnetic torquers are mainly used in low Earth orbit, where the geomagnetic field's intensity is in the order of 25–50 nT (cfr. [Chapter 3](#) – The Space Environment). Rare applications of magnetorquers in high Earth orbits, sometimes up to the geosynchronous altitude [28], are diminished in the available torque values by the low intensity of the Earth's magnetic field, which decreases with the inverse cube of the distance from the center of the Earth. Obviously, magnetorquers cannot be used around celestial bodies without magnetic field, and their application in space regions where the magnetic field is not well understood can lead to unpredictable behavior.

The simplest method to generate a controllable magnetic dipole moment is by exploiting electromagnetic coils according to the Ampère's circuital law, which relates the magnetic dipole value to the current flowing in the coils. An electric current of I amperes along a planar loop of area A produces a magnetic dipole moment with magnitude $m = IA$. The direction of the dipole is perpendicular to the plane of the loop, according to the right-hand rule. If the electromagnetic coils are composed by several loops, they are denoted as solenoids, and the elemental dipoles of the N loops are linearly superimposed resulting in an overall magnetic dipole moment with magnitude.

$$m = N I A. \quad (7.5)$$

The units for the dipole moment are Am^2 . When m is in Am^2 and the magnetic field is specified in Tesla, [Eq. \(7.4\)](#) gives the torque in Nm. The dipole moment formula is a linear function of the current flowing in the electromagnetic coil, the area, and the number of the loops. However, considering realistic current values, large areas or many loops are required to produce the dipole moments needed by the majority of spacecraft. For

for this reason, magnetorquers adopt dedicated construction strategies to achieve acceptable actuation levels.

Magnetorquers assembly

Magnetic torquers assemblies are designed to optimize the dipole moment value for a given amount of current. Three main magnetorquers construction approaches exist:

- Air-core magnetorquers are the immediate application of the basic principle behind electromagnetic coil actuation. A conductive wire is shaped to form multiple loops around a nonconductive support, usually integrated inside or on the side panels of the spacecraft. Eq. (7.4) can be used to compute the exact dipole value for this typology of magnetorquers. Hence, air-core magnetorquers design tends to maximize the available area encompassed by the coil, and the number of superimposed loops.
- Torque rods are the most efficient magnetorquers assemblies. They are composed by a conductive wire wound around a ferromagnetic core which is magnetized when excited by the coil. The core is typically made of materials with very high magnetic permeability (e.g., nickel or cobalt alloys) to reduce power consumption and actuator bulk for a given generated magnetic dipole. The presence of the ferromagnetic element allows to produce a greater dipole moment. However, the magnetization curves of the core material saturate at a low intensity of the applied magnetic field, and they are characterized by both nonlinearities and hysteresis. The latter is responsible for a residual magnetic field that is present even when the coil is turned off, unless a proper demagnetizing procedure is executed. In general, the magnetic dipole moment value of a torque rod is difficult to be accurately predicted given the presence of the ferromagnetic core, whose permeability is dependent from the intensity of the magnetic field. Thus, Eq. (7.4) is not applicable for torque rods, and the cumulative dipole, sum of the solenoid term and the core's magnetization one, shall be computed as:

$$m_{tr} = NIA + V_c M(\mu_r, I, G),$$

where V_c is the volume of the core, and $M(\mu_r, I, G)$ is its magnetization, which is a function of the relative permeability of the material, μ_r , the current intensity in the coil, and the geometry of the core, represented by the geometric parameter G [29]. A dipole curve of a generic

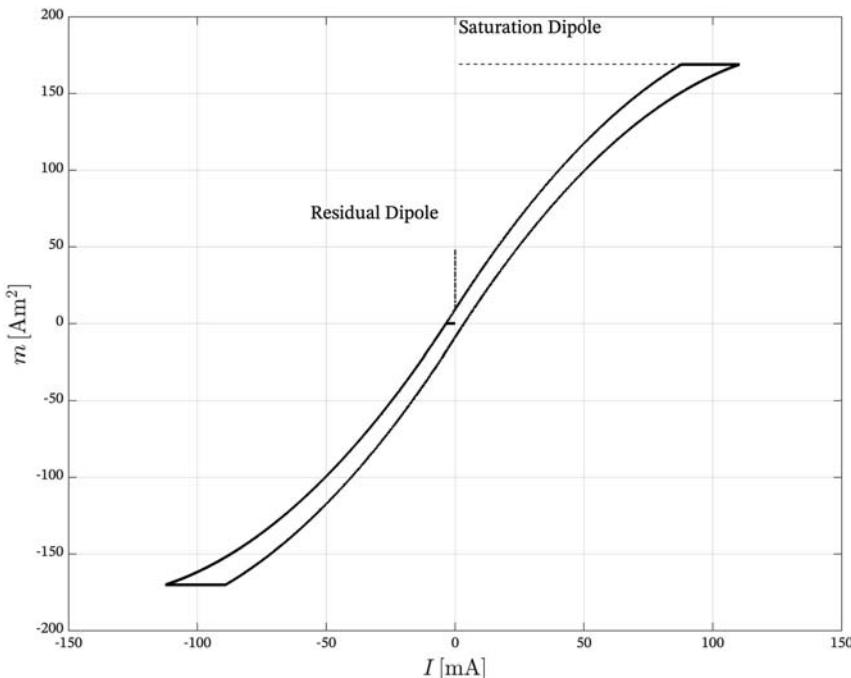


Figure 7.13 Generic torque rod magnetic dipole curve.

torque rod is shown in Fig. 7.13. Commercial torque rods have a residual dipole typically less than 1% of the saturation value, and the linear actuation range is at least 80% of the maximum dipole capacity [30]. Torque rods can commonly generate a dipole of $0.1\text{--}3 \text{ Am}^2$ per 1 mA of current. Note that an air-core would require $NA = 1000 \text{ m}^2$ to achieve a dipole of $1 \text{ Am}^2/\text{mA}$.

- Printed circuit board (PCB)-embedded coils are magnetorquers assemblies fully integrated in the spacecraft components. They are realized creating a spiral trace inside the PCBs of the on-board electronics, typically the solar panels one. The spiral wire is substituted by copper traces in the electronic boards, whose shape is a square spiral on a plane. These magnetorquers are advantageous since their minimal impact on the satellite volumes. However, due to the limits in the board construction and the possible presence of other electronical circuits and components, the maximum magnetic dipole value is severely constrained. Furthermore, the low resistance of the copper traces causes high

current, resulting in a high-power consumption for a given magnetic dipole value. This last drawback can be reduced, thanks to modern PCB technology, which is capable to realize multilayer coil traces inside the electronic boards, increasing the number of available loops for a given area of the board. This specific magnetorquer assembly is suitable for applications in small satellites with stringent mass and volume constraints. Eq. (7.2) is just an approximation of the actual dipole generated by the PCB-embedded magnetorquers. In this case, an average area between the inner and the outer loop of the printed coil, A_{pcb} , and the number of traces forming complete loops, N_{pcb} , shall be used in the magnetic dipole formula. This approximation gives acceptable results as long as the number of spiral traces is not too high (e.g., $N_{pcb} < 40$), otherwise a series term superimposing the Lorentz forces generated by any single segment of the copper traces shall be considered [29].

Magnetorquers actuation function

Magnetorquers are actuated by letting a current flow inside the conductive coils. To achieve the desired dipole moment, the current shall be controlled to coincide with the prescribed value. This is typically achieved with a feedback current control exploiting current measurements from amperometers on the magnetorquer power supply lines. As an alternative, the magnetorquers can be also controlled in terms of supply voltages, for example, by exploiting a PWM voltage regulator. In this last case, the coil resistance shall be known, and, since the resistance of a conductive element varies with the temperature, a voltage control with resistance or temperature feedback may be needed.

In many cases, the actuation laws used on-board shall be as simple as possible. Thus, they are typically based on Eq. (7.5), despite the presence of torque rods with magnetic cores. In very simple applications, the current is not feedback controlled, allowing current fluctuations due to supply voltage or coil temperature variations. In all these cases, the magnetic actuation will be affected by errors with respect to the ideal control law. Hence, the spacecraft GNC shall account for these deviations between computed and actuated torques.

In general, the actuation function of magnetorquers needs to know the local magnetic field, since the magnetic torque is function of both the satellite magnetic dipole and the surrounding magnetic field. Thus, a spacecraft with magnetic actuators shall be equipped with magnetometers to guarantee

the on-board magnetic measurements, as discussed in [Chapter 6 – Sensors](#). Particular attention shall be paid to avoid interferences between the control action of magnetorquers and the magnetometers measurements.

Magnetorquers performance

Typically, magnetorquers applications use three torquers producing magnetic dipole moments on orthogonal axes. Additional magnetorquers can be used to provide additional actuation capabilities, given that the total magnetic dipole is a vector sum of the contributions of the single torquers. Instead, redundancy is commonly guaranteed inside any single torquer, thanks to dual or multiple independent conductive windings. The set of magnetic torquers used in a spacecraft can combine the different construction approaches described before. A typical solution is composed by an air core torquer, with two orthogonal torque rods lying in the plane of the large coil, as seen in [Fig. 7.14](#). In this case, the magnetic dipole of the air core is along one axis orthogonal to the ones of the torque rods. The dipole moments of the three elements have the same order of magnitude because the large area of the air core torquer balances the ferromagnetic material inside the torque rods. Very small spacecraft may use permanent magnets in place of electromagnetic coils to achieve a passive stabilization with respect to the surrounding magnetic environment.

Typical applications of magnetorquers are initial satellite spin dumping, also known as detumbling, and reaction wheels or CMGs stored

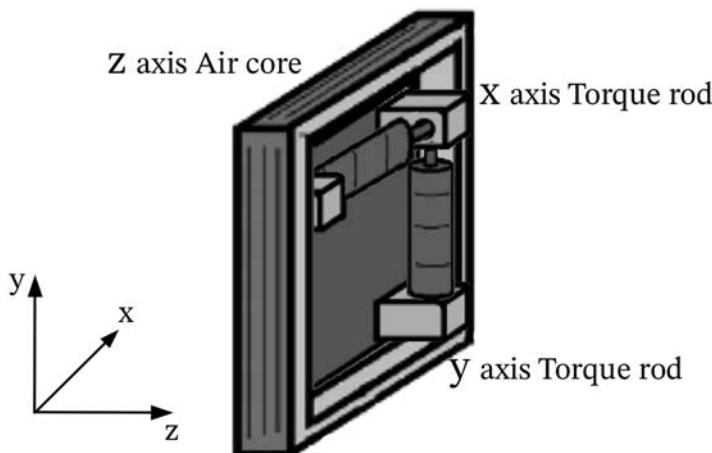


Figure 7.14 Common three magnetorquers set.

momentum dumping. In these cases, magnetic torques are particularly effective in generating a dissipative torque capable to dump the angular momentum. They can be also conveniently used as a redundant actuation system, in case of a failure in the primary actuators. Small and very simple satellites may even use magnetorquers as primary actuators for attitude stabilization and pointing. However, the available performance cannot be compared to other actuation methods. In fact, a significant disadvantage is that the magnetic torques are constrained to lie in the plane orthogonal to the magnetic field, as is evident from Eq. (7.4), so only two out of three axes can be controlled at a given time instant. Full three-axis controllability is potentially available — in an integral sense — over a complete orbit. However, this is only true if the spacecraft’s orbital plane does not coincide with the geomagnetic equatorial plane and does not contain the magnetic poles [31]. The geomagnetic field irregularities and its rotation with the Earth introduces many harmonics in the magnetic field evolution along the orbit. Thus, simulations involving magnetic control should be at least 24 h in length, ensuring that all the frequencies of the magnetic environment harmonics are accounted in the control design [32]. Undesirable magnetic dipoles inside the spacecraft lead to magnetic disturbance torques, and, even if they are generally few orders of magnitude smaller than the magnetorquers ones, they shall be considered as well in the magnetic control design.

Notwithstanding the problems and limitations, magnetorquers have many advantages. They are lightweight; they do not require any propellant consumption; they produce a pure torque, without spurious forces perturbing the spacecraft’s orbit; they have no moving parts, resulting in a high reliability and eliminating internal microvibrations. Their typical applications in near-Earth orbits allow to produce magnetic control torques ranging from $2 \cdot 10^{-6}$ to 0.05 Nm, since commercially available torquers provide dipole moments from 0.1 to 1000 Am². Table 7.7 resumes strengths and weaknesses of magnetorquers.

Table 7.7 Magnetorquers strengths and weaknesses.

Strengths	Weaknesses
Lightweight	Small torques
Energy efficient	Torque constrained to lie in a plane
Reliable	Required presence of an external magnetic field
No orbital perturbation or microvibrations	Instantaneous three-axis control not possible
Efficient in generating dissipative torques	

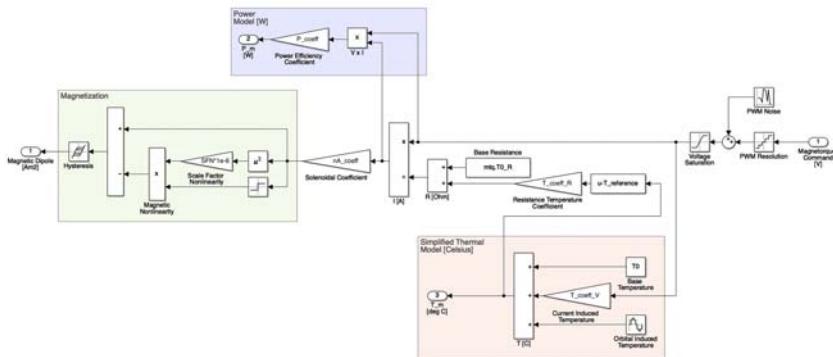


Figure 7.15 Example magnetorquer model.

Magnetorquers model

An example magnetorquer model implementation in MATLAB/Simulink is reported in Fig. 7.15. The input command is a PWM voltage, including resolution, saturation, and noise effects. The voltage applied to the electric coil composing the magnetorquer results in a current, which is a function of the temperature-dependent magnetorquer's resistance. A simplified thermal model of the actuator is included to represent the resistance variation with the temperature. The latter has a periodic oscillation with the orbital period, and it is dependent from the actuation intensity. Finally, the magnetization of the magnetorquer's core is modeled including the hysteresis and the magnetic nonlinearity. The power consumption model is proportional to the input voltage and flowing current.

References

- [1] F.L. Markley, J.L. Crassidis, Fundamentals of Spacecraft Attitude Determination and Control, STL, 2015.
- [2] V.J. Modi, K. Kumar, Attitude control of satellites using the solar radiation pressure, *Journal of Spacecraft and Rockets* 9 (9) (1972) 711–713.
- [3] Les Johnson, et al., Solar sails: technology and demonstration status, *International Journal of Aeronautical and Space Sciences* 13 (4) (2012) 421–427.
- [4] R. Sun, et al., Roto-translational spacecraft formation control using aerodynamic forces, *Journal of Guidance, Control, and Dynamics* 40 (10) (2017) 2556–2568.
- [5] K.C. Pande, R. Venkatachalam, On optimal aerodynamic attitude control of spacecraft, *Acta Astronautica* 6 (11) (1979) 1351–1359.
- [6] H.A. Talebi, K. Khorasani, T. Siamak, A recurrent neural-network-based sensor and actuator fault detection and isolation for nonlinear systems with application to the satellite's attitude control subsystem, *IEEE Transactions on Neural Networks* 20 (1) (2008) 45–60.

- [7] A. Rahimi, K. Dev Kumar, H. Alighanbari, Fault detection and isolation of control moment gyros for satellite attitude control subsystem, *Mechanical Systems and Signal Processing* 135 (2020) 106419.
- [8] I. Gueddi, et al., Fault detection and isolation of spacecraft thrusters using an extended principal component analysis to interval data, *International Journal of Control, Automation and Systems* 15 (2) (2017) 776–789.
- [9] M.J.L. Turner, *Rocket and Spacecraft Propulsion: Principles, Practice and New Developments*, Springer, 2009.
- [10] W. Emrich Jr., *Principles of Nuclear Rocket Propulsion*, Elsevier, 2016.
- [11] G. Garbe, An overview of NASA’s solar sail propulsion project, in: 39th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, 2003.
- [12] J.R. Wertz, J. Wiley, Larson, *Space Mission Analysis and Design*, third ed., Space Technology Library, Springer, 2005.
- [13] W. Fehse, *Automated Rendezvous and Docking of Spacecraft*, Cambridge University Press, 2003.
- [14] W. Press, S. Teukolsky, W. Vetterling, B. Flannery, *Numerical Recipes in C*, Cambridge University Press, Cambridge, UK, 1992.
- [15] B. Armstrong-Helouvry, *Control of Machines with Friction*, Kluwer, Boston, MA, 1992, 1991.
- [16] L. Márton, B. Lantos, Modeling, identification, and compensation of stick-slip friction, *IEEE Transactions on Industrial Electronics* 54 (1) (2007).
- [17] D. Karnopp, Computer simulation of slip-stick friction in mechanical dynamic systems, *Journal of Dynamic Systems, Measurement, and Control* 107H11 (1985) 100–103.
- [18] ISO 21940-11:2016 – Mechanical Vibration — Rotor Balancing — Part 11: Procedures and Tolerances for Rotors with Rigid Behaviour.
- [19] ISO 1940-1:2003 – Mechanical Vibration — Balance Quality Requirements for Rotors in a Constant (Rigid) State — Part 1: Specification and Verification of Balance Tolerances.
- [20] H. Kurokawa, Survey of theory and steering laws of single-gimbal control moment gyros, *Journal of Guidance, Control, and Dynamics* 30 (5) (2007) 1331–1340.
- [21] G. Margulies, J.N. Aubrun, Geometric theory of single-gimbal control moment gyro systems, *Journal of the Astronautical Sciences* 26 (2) (1978) 221–238.
- [22] B. Wie, Singularity analysis and visualization for single-gimbal control moment gyro systems, *Journal of Guidance, Control, and Dynamics* 27 (2) (2004) 271–282.
- [23] L. Jones, R. Zeledon, M.A. Peck, Generalized framework for linearly constrained control moment gyro steering, *Journal of Guidance, Control, and Dynamics* 35 (4) (2012) 1094–1103.
- [24] B. Wie, Singularity escape/avoidance steering logic for control moment gyro systems, *Journal of Guidance, Control, and Dynamics* 28 (5) (2005) 948–956.
- [25] B. Wie, *Space Vehicle Dynamics and Control*, second ed., AIAA, Ames, 2008.
- [26] F.A. Leve, B.J. Hamilton, M.A. Peck, *Spacecraft Momentum Control Systems*, Springer International, 2015.
- [27] R. Votel, D. Sinclair, Comparison of control moment gyros and reaction wheels for small earth-observing satellites, in: *Proceedings of the 26th Annual AIAA/USU Conference on Small Satellites*, 2012.
- [28] D. Desiderio, M. Lovera, S. Pautonnier, R. Drai, Magnetic momentum management for a geostationary satellite platform, in: *Proceedings of the 47th IEEE Conference on Decision and Control*, Cancun, 2008, pp. 1243–1248.
- [29] N. Bellini, *Magnetic Actuators for Nanosatellite Attitude Control*, Alma Mater Studiorum Università di Bologna, 2014.

- [30] F. Landis Markley, J.L. Crassidis, Fundamentals of Spacecraft Attitude Determination and Control, Space Technology Library: Springer, 2014.
- [31] S.P. Bhat, A.S. Dham, Controllability of spacecraft attitude under magnetic actuation, in: Proceedings of the 42nd IEEE Conference on Decision and Control, Maui, 2003, pp. 2383–2388.
- [32] A. Colagrossi, M. Lavagna, Fully magnetic attitude control subsystem for picosat platforms, *Advances in Space Research* 62 (12) (2018) 3383–3397.

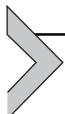
This page intentionally left blank



PART TWO

Spacecraft GNC

This page intentionally left blank



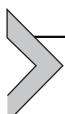
Guidance

Thomas Peters¹, Stefano Silvestrini², Andrea Colagrossi²,
Vincenzo Pesce³

¹GMV Aerospace & Defence, Madrid, Spain

²Politecnico di Milano, Milan, Italy

³Airbus D&S Advanced Studies, Toulouse, France



What is guidance?

Since the early ages of humanity development, the first activity that has engaged individuals is to generate plans. Let us imagine a person who leaves the shelter to fulfill the objective of “living life.” The first decision to be made is the selection of a desired place to visit. It can either be a forest to go hunting, for our primitive ancestors, or to the office, in our daily life, or simply a nice place to be in, such as a mountain, a park, or a beach. Immediately after, the person would need to think of a way to get to the target place by planning the journey. Generally, people would think of intermediate places they expect to visit while being on-track on their way. At this stage, it does not really matter what can happen in between. This simple example illustrates the concept of *guidance* in its pure essence. Although the difference between guidance and control is sometimes blurred, the research community agrees that the act of planning the places to visit and generating the desired path between them is what we refer as guidance. A good way to distinguish the domains is to think of guidance as an activity performed beforehand that does not necessarily take into account unforeseen events that may happen.

Nevertheless, as real life is truly complicated, this is not always true. The guidance concept referring to the generation of paths to follow can be updated as our journey prosecutes. Let us go back to the example of a person “living life”: let us assume that the individual wanted to reach the desired place using a bicycle; unfortunately, the bike has a flat tire, thus the person is forced to take the car. In the original plan, the human being would have cycled through the center of the city, but this is no longer possible with the car due to circulation restrictions in the old town. The individual, while making the decision of changing the transportation system, regenerates a

plan according to such unexpected event. This is what is often referred as *adaptive guidance*, meaning that the agent is capable of reshaping its plans according to what it is currently experiencing.

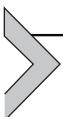
In technical terms, guidance refers to the determination of the desired travel path, commonly referred as trajectory in the space domain, from the spacecraft current state to a designated target one, being either a position, an orientation, or a given orbit. Guidance entails also the desired changes in velocity necessary to follow the defined path.

The interface with the other core modules of navigation and control is quite intuitive. The guidance system generates the intended plan, which is constantly compared to the navigation output to let the system understand whether it is in its correct state at the current time. The control uses such comparison to synthesize the actions to correct any, inevitable, mismatch between where we would like the spacecraft to be and where it actually is. On the other hand, especially in adaptive applications, the guidance interrogates the navigation to acquire the current state to generate new plans based on the real status of the journey.

With these concepts in mind, we have all the tools to start diving into the technicalities of the guidance system and algorithms.

This chapter is composed by the following sections:

- *On-board versus ground-based guidance.* The guidance system is the formal discriminant between attitude and orbit control system (AOCS) and guidance, navigation, and control (GNC). This section explores the differences in the implementation philosophies.
- *Guidance applications.* This section explores the guidance system design process from the understanding of the dynamical system to the description of different guidance techniques. It covers critical topics such as optimal control, interpolation, and two applicative cases, namely rendezvous guidance and attitude guidance.
- *Guidance implementation best practices.* A series of best practice tips and checks is collected in this section, giving a more practical perspective on the implementation of guidance algorithms.



On-board versus ground-based guidance

A space mission consists of the ground segment and the space segment. The space segment is the spacecraft, which itself consists of the spacecraft bus and the payload. Fig. 8.1 shows these elements of the space system and the relations between them. Controlling and operating a spacecraft is a task

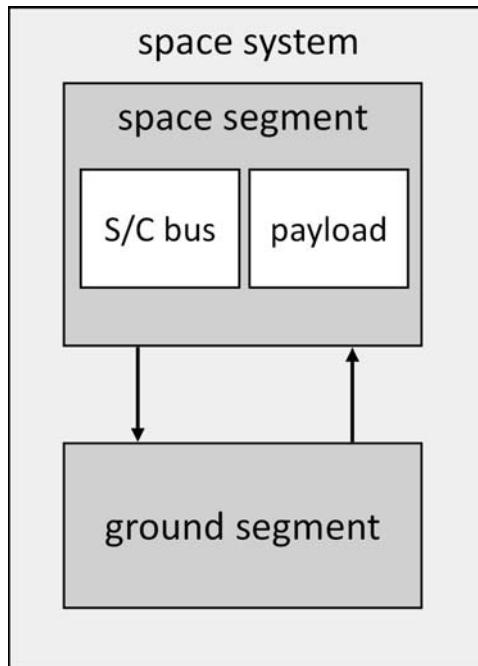


Figure 8.1 Elements of a space system.

shared between the ground segment and the space segment. The level of on-board autonomy determines the capabilities of the space segment to control its own orbit and attitude to fulfill the mission needs.

In general, as described in [Chapter 1](#) — Introduction, on-ground activities rely on significantly higher computational power compared to on-board resources. Specific to the guidance system, this means that the ground segment can use more complete models of the spacecraft dynamics and sensors, more sophisticated filtering techniques to obtain a guidance solution, and more detailed dynamics models for computing maneuvers. Maneuvers can be computed on ground using optimization techniques that may be prohibitively expensive to implement on-board or using techniques that require human supervision to ensure that the maneuvering solution is acceptable. Moreover, the capabilities guidance systems based on the on-ground segment is highly dependent on the number of ground stations, ground link frequencies, and delays. For instance, rendezvous missions may require autonomous abort capabilities for the close proximity operations that may not be achievable with ground-based systems [\[1\]](#).

As discussed in [Chapter 1](#) – Introduction, the increased level of autonomy also requires more time and effort to be spent on the development, verification, and validation of the software, which increases the cost of the on-board software. Nevertheless, it is safe to claim that the reduction of the cost of operation is expected to be greater than the increased cost of the software development [2].

For the sake of completeness, referring to [Chapter 1](#) – Introduction, we report again the nomenclature definition of on-board autonomy. The European ECSS Space Segment Operability Standard provides a definition of on-board autonomy [3]: “*On-board autonomy management addresses all aspects of on-board autonomous functions that provide the space segment with the capability to continue mission operations and to survive critical situations without relying on ground segment intervention.*”

Finally, the reasons for increasing the level of on-board autonomy are the following, which are coherent with what explained for the entire GNC system in [Chapter 1](#) – Introduction:

- It reduces the complexity of ground operations.
- It implies that the ground segment no longer has direct control over on-board operations that may be critical to the survival of the spacecraft [4].
- It reduces personnel requirements on ground (possibly single-shift), may reduce propellant cost and allows for more precise timing of on-board events [2].
- It avoids limited communication windows and/or large communication delays, such as, for example, planetary sample return missions.
- It reacts promptly to unexpected events, such as, for example, detection of an imminent collision and trajectory replanning.

The guidance function often plays a key role in the development of on-board autonomy, as the guidance function is in charge of defining the behavior of the system at medium to long time scales. The guidance can abstractly define plans (here defined as a sequence of maneuvers and trajectories with a clearly defined objective to change the state of the space system into another desired state) that determine what actions the space segment will perform. The guidance can encode alternative plans and recovery plans that can be activated when a change in the environment is detected that calls for a different course of action than what was encoded in the original plan. The guidance performs an abstraction step, encoding specific maneuvers and trajectories as more abstract elements in a set of available behaviors of the space system as a whole (see also Section [Guidance modes](#)). Additional layers of abstract reasoning can be added that interact with the guidance (and the

navigation) to ensure that the system can adapt to changing situations autonomously.



Guidance applications

This section covers the technicalities of the development of a guidance system. The design process is described, mentioning the most common techniques to define the guidance system and algorithms. Moreover, the interaction between the guidance module and the spacecraft at system level is investigated, from requirements definition to architectures.

Design process

This section describes the technical design process of the guidance system. It entails the critical steps to develop effective and successful algorithms, from the understanding of the dynamical system to the core concept of optimization. Finally, two guidance applicative cases are described to demonstrate how the theoretical concepts turn into practice.

General design approach

The objectives of the guidance design can be different for different phases of the design process. During the initial phases of the mission design, the tasks of the guidance are closely related to the mission analysis. Feasible trajectories that satisfy the objectives are defined, and the ΔV 's that are required to follow the trajectory are calculated to establish a preliminary ΔV budget. These ΔV 's are in some sense ideal because the calculation does not take into account the navigation uncertainty or the actuation errors. The effects of these uncertainties and errors are absorbed into margins on the ΔV budget.

Furthermore, during the initial stage of the design, the needs and constraints of the system are explored, and these are eventually encoded into requirements on the design of the system. For the guidance functions, the development focuses on lower-level utility functions that can be tested independently and early in the process. These functions may be reused from other projects, and specific special purpose functions may be built that cover the peculiarities of the specific mission. For example, for rendezvous and formation flying, function libraries that can be reused may include free-flying trajectory propagation functions, impulsive maneuver computation functions, forced motion, and attitude pointing functions.

The definition of guidance functions needs to start with a fundamental understanding of the dynamics. The dynamics of rendezvous from the point

of view of the development of a guidance function is discussed in Section [Understanding the dynamical system](#), as well as algorithms for relative trajectory propagation. Algorithms for impulsive, forced motion, and attitude maneuver computation functions are explored in Sections [Impulsive maneuvers and trajectories](#), [Forced motion](#), and [Application: Attitude guidance](#). Section [Design of a guidance function](#) briefly discusses how and what type of requirements may be defined for a guidance function.

Understanding the dynamical system

To start the design of the guidance function, it is crucial to develop an understanding of the dynamics and the operating state of the system for which the guidance function is developed. Mission analysis leads to the identification of the principal dynamical effects and disturbances to take into account into the dynamical model.

From a practical point of view, the selection of the level of accuracy of the dynamical model for the guidance depends on a number of factors. The most important of factors are: short-term versus long-term propagation, available computational power, navigation and actuation accuracy, and the frequency of maneuver application. Consistency of the dynamical model between the GNC can be beneficial from the perspective of verification and validation, especially if the exact same implementation of the dynamics is used. If the maneuvers are spaced close together, then the propagation arcs are short, and lower accuracy models can be sufficient. For on-board applications, computational resources may be limited, and this may drive toward using lower-accuracy dynamical models within the guidance. If the navigation accuracy is low, then the guidance cannot expect to provide high-accuracy results and the dynamical model used in the guidance can be rather simple. Lastly, if the maneuver application errors are expected to be large, then it is not necessary to compute highly accurate maneuvers, and the guidance dynamics models can be simplified. The guidance function often needs to be able to handle different levels of navigation accuracy in addition to the peculiarities of the control function and the actuators. Recent articles [5] have suggested taking a mixed approach. In this approach, the guidance first computes the desired reference trajectory during prespecified intervals with discrete changes at the boundaries of these intervals. A high-accuracy dynamical model is used to propagate the desired reference trajectory during these intervals. A lower accuracy dynamical model is used to compute the maneuvers required to perform the changes at the boundaries of the intervals.

Guidance representations

The representation of the state vector within the guidance software depends on the needs of other subsystems, on the comprehensibility to human users that check and verify the software, and possible performance impacts of the representation. From the point of view of verification and validation, it is desirable to use the same representation for the GNC functions. As a minimum, it must be ensured that the interfaces between these elements use the exact same representation of the state vector.

The guidance is responsible for providing the reference trajectory to the other GNC functions. A wide variety of models and representations is available, and it is the task of the GNC engineer to select the most suitable representation of the reference trajectory for the application at hand. It may be desirable to have a consistent formulation of the dynamical model between the GNC functions, although the internal representation inside each of these functions can be different from the output representation. A common approach for translational dynamics is to make use of a full nonlinear reference trajectory and a state transition matrix to compute small trajectory corrections close to the reference. In this case, it is desirable to ensure that the dynamical model used for the full nonlinear trajectory is consistent with the linear model used for the state transition matrix, at least for the major dynamical effects. The need for linearization may also form an important driver in the selection of parameters to represent the trajectory. The kinematics of certain sets of parameters can be affected by nonlinear effects to a greater or lesser extent. In other cases, the transformation between certain representations may suffer more from nonlinear effects, while the dynamics are affected less. A judicious choice of parameters and a careful consideration of what aspects of the dynamical system to linearize can lead to great improvements in the accuracy of models. See Ref. [6] for a thorough discussion of nonlinear effects in both orbital and attitude dynamics.

For attitude guidance, the following representations can be considered (cfr. [Chapter 5 – Attitude](#)):

- Direction cosine matrix. The direction cosine matrix provides an unambiguous and easy to understand representation of the rotation that is free from singularities, but it uses many more parameters than are strictly required to provide the attitude information.
- Euler angles. The Euler angles are another example of an easy-to-understand representation of the attitude information that uses the minimum number of parameters (3). Difficulties arise because the Euler angles contain singularities and because multiple (12) sets of Euler angles exist.

Care must be taken firstly to avoid these singularities by choosing the right operating point and secondly to use the same definition of the angles throughout the system.

- Axis-angle and Euler vector. The axis-angle and Euler vector systems are variations of the same idea. In the case of axis-angle representation, the rotation axis \mathbf{e} and the rotation angle ϑ are provided to unambiguously represent the attitude. In the case of the Euler vector, the rotation axis is multiplied by the rotation angle. In this case, the topology of the space can be represented by a ball of radius π in which opposite points on the sphere are identified.
- Quaternions. Unit quaternions (quaternions of length one) are commonly used to represent rotations. Quaternions contain a scalar part q_4 equal to the cosine of half the rotation angle, and a vector part $\mathbf{q}_{1:3}$ equal to the sine of half the rotation angle times the rotation axis. The advantage of quaternions is that they require a minimum number of operations to perform the rotation operation, and only multiplication, addition, and subtraction are used. The downside of quaternions is that they are more difficult to understand. In addition, there is some ambiguity in the definition of the order of the elements of the quaternion: scalar first or scalar last.
- Rodrigues parameters/Gibbs vector. The Rodrigues parameters or Gibbs vector is a three-element vector that is obtained by dividing the vector part of the quaternion by the scalar part of the quaternion, $\frac{\mathbf{q}_{1:3}}{q_4}$. This means that the Rodrigues parameters are equal to the tangent of half the rotation angle times the rotation axis. The Rodrigues parameters are singular when the rotation angle is equal to π .
- Modified Rodrigues parameters. The modified Rodrigues parameters are a stereographic projection of the sphere of unit quaternions onto a three-dimensional plane. This leads to the following expression for the modified Rodrigues parameters $\mathbf{p} = \frac{\mathbf{q}_{1:3}}{1+q_4}$. The modified Rodrigues parameters are equal to $\mathbf{e} \tan \frac{\vartheta}{4}$. The modified Rodrigues parameters move the singularity to a rotation through an angle of 2π . This is not necessarily a problem as a rotation through an angle of 2π is equivalent to the rotation through an angle 0.

A comprehensive survey of attitude representations can be found in Ref. [7]. This reference provides kinematics equations for all attitude representations listed above and includes a discussion of attitude errors.

Reference [8] (page 181–183) provides a topological analysis of the Euler angles to demonstrate the location and the nature of the singularities. References [7,9] provide a full list of all possible combinations of Euler angles, and Ref. [9] provides some options and considerations for handling or avoiding singularities.

Useful references for the unit quaternions are [10,11]. Ref. [11] uses a slightly different notation for the quaternions but provides useful insight into the construction of rotation sequences using quaternions. Ref. [10] provides extensive discussions on quaternions and their relationship to rotation matrices. This reference is mainly focused on navigation filters, but it contains useful ideas and mathematical expressions for attitude guidance development.

Refs. [7,12] provide discussions of the Rodrigues parameters and the modified Rodrigues parameters. Ref. [12] additionally provides geometrical insight into how the Rodrigues parameters are obtained and provides several useful generalizations.

A wide variety of parameterizations are available for modeling orbital motion.

- Cartesian state. The inertial Cartesian state is easily understood and can be used directly to perform geometrical calculations (such as, for example, calculation of direction vectors or range) in a straightforward manner. The inertial Cartesian state can be used to model the orbital motion directly, but there are some disadvantages with respect to orbital elements. If a numerical integration scheme is used, then the step size generally needs to be smaller for the Cartesian state than for other representation. The values of the components of the Cartesian state tend to vary more rapidly than orbital elements, such that more parameters may be required to represent the evolution of the trajectory.
- (Modified) Keplerian orbital elements. The (modified) Kepler elements have an easy geometrical meaning and are treated in nearly all introductory texts on astrodynamics. In an unperturbed orbit, the orbital elements are constant, and in a perturbed orbit, the elements tend to vary slowly. This makes the orbital elements easy to represent using a relatively small number of parameters. This advantage applies to all orbital element sets. The main disadvantage of the Keplerian orbital elements is that singularities exist for zero eccentricity and zero inclination. Despite this disadvantage, the Keplerian elements are commonly used to represent orbit information.

- Equinoctial elements. The set of equinoctial elements is nonsingular for small eccentricities and inclinations. The equinoctial elements can easily be obtained from the (modified) Keplerian elements, but the meaning of each of the individual elements is a little harder to understand when compared to the Kepler elements.
- Modified equinoctial elements. The set of modified equinoctial elements is nonsingular for all eccentricities and inclinations. This set is therefore suitable for any orbit. In particular, the modified equinoctial elements are commonly used for low-thrust escape trajectories.
- Delaunay elements. The Delaunay elements are a canonical set of orbital elements. This means that the variational equations take on a particularly simple form and perturbation calculations are greatly simplified.
- Poincaré elements. The Poincaré elements are a canonical set of elements that is nonsingular for small eccentricities and inclinations.

Ref. [13] provides an overview of the different sets of orbital elements and a discussion of the advantages and disadvantages of each of these sets including the availability of the variational equations for each of these sets. A distinction needs to be made between the internal representation of the guidance reference and the representation of the orbit data that the guidance provides to other functions. It can be beneficial to internally represent the reference orbit in terms of one of the many sets of orbital elements and present the Cartesian state to other functions. In this case, the guidance needs to contain transformation functions that can transform between different representations of the orbit.

Linearized relative motion models have many different applications including the launch vehicle guidance, midcourse correction maneuver calculation for interplanetary trajectories, orbit control and formation flying, and rendezvous. In recent years, rendezvous and formation flying have driven the development of many different theories for relative motion. Reference [14] provides a recent survey of relative motion models and includes flow diagrams that explain how these models work. One key observation is that while linearized orbital elements themselves do not suffer greatly from errors due to linearization, the transformation between Cartesian state vector and orbital elements itself does. This insight has led to the development of theories that use a full nonlinear transformation between the Cartesian state vector and the relative orbital elements, and a linear dynamics model for the relative orbital elements. These models combine the advantages of the linear models with the accuracy provided by using the nonlinear transformations.

Optimization

Many guidance problems attempt to formulate an optimal planning for the system to be controlled. The optimal planning minimizes or maximizes some form of cost function that can be expressed as a function of the state variables, the control inputs, and/or the time. The guidance provides a planning that maximizes the performance of the system to be controlled or that minimizes the amount of resources needed to follow such a plan. Some examples of optimization problems that occur in space engineering are the following:

- To find the steering program that maximizes the amount of payload a launch vehicle can deliver to a specified orbit.
- To find the burn program that maximizes the change in semimajor axis for a spacecraft with a single electric propulsion thruster operating with constant thrust.
- To find the sequence of impulsive maneuvers that minimizes the amount of propellant required to perform an orbit transfer.
- To find the steering program for reaction wheels that minimizes the amount of energy used to perform a slew maneuver.

Aerospace optimization problems often feature constraints, which may either be equality constraints or inequality constraints. Initial and terminal conditions can be viewed as equality constraints, and so can the dynamical equations in general. Some examples of inequality constraints are:

- The dynamical pressure on a launch vehicle must remain below a certain value.
- The angle between the boresight of a star tracker and the Sun direction must remain greater than a certain value during slew maneuvers.

Many optimization problems that occur in space engineering are nonlinear and require iterative methods to solve. This poses several problems for on-board implementation. Firstly, the methods can be computationally expensive, and this can be a problem for space-qualified processors that generally have less processing power than on-ground processors. Secondly, the intermediate solutions provided by the algorithm may not satisfy all the constraints, and this means that the intermediate solutions are not feasible. Thirdly, the iterative method may not converge to a (globally) optimal solution at all. In such cases, it may be preferable to rely on off-line trajectory planning methods that can use ground-based computational resources and operator supervision to check the correctness and feasibility of the solution.

The solution of the optimal planning problem consists of the time history of the state vector and the control variables. This approach highlights a

difference between the traditional objectives of a guidance function versus the objectives of a control function. The guidance function provides an open-loop feedforward trajectory, while the control function constructs the feedback required locally to drive the solution to the desired terminal point. Fig. 8.2 illustrates this idea. The guidance function constructs an optimal time history of the control variables, whereas the control function constructs optimal feedback that is applicable to each point in the domain.

Optimization methods typically have more free parameters than are required by the degrees of freedom that are determined by the (equality) constraints. The problem is typically underconstrained, and this freedom is exploited to search for an optimum. For example, in a fixed-time orbit transfer problem, there are three constraints for the initial position and three for the terminal position, while the time of flight is fixed. In this case, the six parameters that form the solution (2×3 , namely the initial and the terminal velocity) are fully determined. If the initial and/or the terminal time are left free, then the total ΔV can potentially be minimized by changing the maneuver times.

Classical formulation of the optimal control problem

Optimization generally involves iteratively approximating the cost function by a quadratic function and finding the minimum of this approximation. A case of one-dimensional minimization is illustrated in Fig. 8.3. The unconstrained minimum occurs when the first derivative of the function is equal to

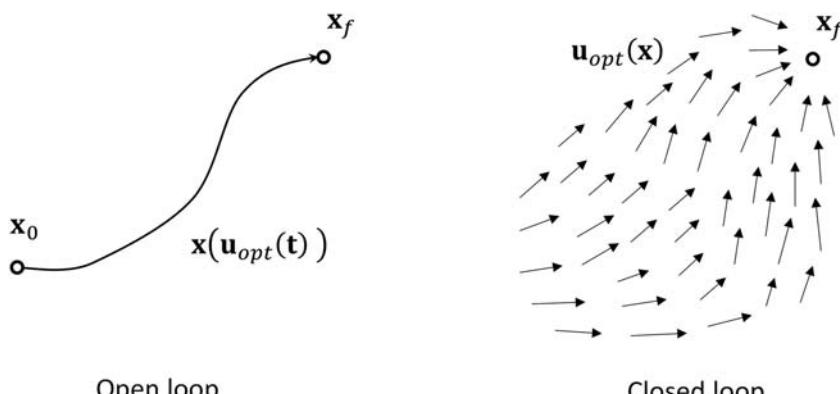


Figure 8.2 Open-loop, feedforward optimization versus closed-loop, feedback optimization. Adapted from M. Kelly, *An introduction to trajectory optimization: how to do your own direct collocation*, SIAM Review 59 (4) (2017) 849–904.

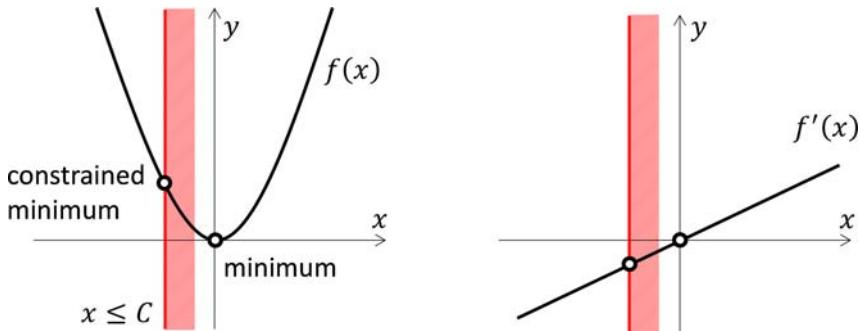


Figure 8.3 The optimum corresponds to a zero of the first derivative of the function f .

zero. The figure also shows what happens if a constraint is present. In this case, the constraint is $x \leq C$. The minimum lies on the constraint and the constraint is active. Note that if the constraint would be $x > C$, then the original minimum would be found, and the constraint would be inactive.

In the classical development of optimization of a dynamical system, the calculus of variations is used [15]. The cost function can be presented in the Lagrange, Bolza, or Mayer form. These formulations are equivalent [16]. The development of the optimal control by Ref. [17] is followed here. The system is described by the following differential equations:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$$

The cost functional is presented in Bolza form as follows:

$$J = \phi(\mathbf{x}_f, t_f) + \int_{t_0}^{t_f} L(\mathbf{x}, \mathbf{u}, \tau) d\tau$$

In this equation, ϕ is the terminal penalty term and L is the Lagrangian term of the cost function (see Ref. [17], page 50). The dynamics equations are treated as constraints in the optimization function. The classical approach to this problem appends the constraints to the cost function using a vector of Lagrange multipliers $\boldsymbol{\lambda}$.

$$\bar{J} = \phi + \int_{t_0}^{t_f} \left[L + \boldsymbol{\lambda}^T (\mathbf{f} - \dot{\mathbf{x}}) \right] d\tau$$

Define the Hamiltonian $H = L + \boldsymbol{\lambda}^T \mathbf{f}$ and integrate by parts to obtain the following expression:

$$\bar{\bar{J}} = \phi - \boldsymbol{\lambda}_f^T \mathbf{x}_f + \boldsymbol{\lambda}_0^T \mathbf{x}_0 + \int_{t_0}^{t_f} \left[H + \dot{\boldsymbol{\lambda}}^T \mathbf{x} \right] d\tau$$

The next step is to take the first variation of this expression. The first variation examines the behavior of solutions close to the current solution. In other words, the first variation examines how the cost function changes because of small changes in the state vector and the control variables. These small changes in the state or the control are expressed by introducing the symbol δ . That is to say, a small change in, for example, the control inputs is expressed as $\delta \mathbf{u}$.

$$\delta \bar{\bar{J}} = \left(\frac{\partial \phi}{\partial \mathbf{x}_f} - \boldsymbol{\lambda}_f^T \right) \delta \mathbf{x}_f + \boldsymbol{\lambda}_0^T \delta \mathbf{x}_0 + \int_{t_0}^{t_f} \left[\left(\frac{\partial H}{\partial \mathbf{x}} + \dot{\boldsymbol{\lambda}}^T \right) \delta \mathbf{x} + \frac{\partial H}{\partial \mathbf{u}} \delta \mathbf{u} \right] d\tau$$

For an optimal solution, $\delta \bar{\bar{J}} = 0$. From this condition, the following set of equations are obtained.

The adjoint equations:

$$\dot{\boldsymbol{\lambda}}^T = - \frac{\partial H}{\partial \mathbf{x}}$$

The transversality or boundary conditions:

$$\boldsymbol{\lambda}_f^T = \frac{\partial \phi}{\partial \mathbf{x}_f}$$

$$\frac{\partial \phi}{\partial t_f} + H_f = 0$$

The control equations:

$$\frac{\partial H}{\partial \mathbf{u}} = \frac{\partial L}{\partial \mathbf{u}} - \boldsymbol{\lambda}^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = 0$$

These are *necessary* conditions for an optimum to occur. To obtain the sufficient conditions for an optimum, the second variation needs to be examined. In this case, the Hessian matrix (the matrix of second derivatives) of the cost function and the constraints needs to be positive semidefinite.

The classical approach using calculus of variations has some drawbacks, and modern approaches to optimization and optimal control use a somewhat different approach (see Ref. [18]).

Indirect methods versus direct methods

Optimization techniques are generally divided into two categories, direct and indirect methods. Typically, direct methods attempt to construct a sequence of approximations to the optimal solution such that the cost function is minimized. Given a generic cost function $F(x)$, a direct optimization algorithm iteratively constructs a sequence of points x_1, x_2, \dots, x_{opt} such that $F(x_1) > F(x_2) > \dots > F(x_{opt})$. Indirect methods on the other hand operate by finding a root (that is, a zero crossing) of the necessary conditions for an optimum, $F'(x) = 0$. Direct methods generally do not require the formulation and solution of the adjoint equations, the control equations, or the transversality conditions.

Trajectory optimization methods

Many different methods are in use for performing trajectory optimization. There are several introductory books that provide in-depth descriptions of the most common methods, such as, for example, Refs. [19,20]. Some of the more common trajectory optimization methods are:

- Primer vector. The primer vector method integrates the adjoint equations to find the optimal control inputs. The primer vector method is often used to find optimal low thrust and impulsive transfers. See Refs. [19,21,22] for more details on this method.
- Collocation methods. Collocation methods determine an approximate solution by enforcing the condition that the dynamics equations are satisfied at certain given points called the collocation nodes or points. Collocation methods typically discretize time to produce a mesh, and a parameterization for the state vector and the control variables (for example, piecewise polynomials) for each segment. Ref. [19] provides an example of trajectory optimization using a collocation method that uses Hermite polynomials to represent the state vector history and the control variables.
- Pseudospectral methods. Pseudospectral methods use orthogonal basis functions to parameterize the trajectory. An example of basis functions are the Chebyshev polynomials. The solution is found by finding the derivatives of the trajectory in terms of the parameters and ensuring that the derivatives agree with the differential equation at a specified set of nodes.

In contrast to the collocation method which uses basis functions valid for each segment, the pseudospectral methods tend to use global basis function over the entire interval [19,20,23].

- Shooting method. The shooting method iteratively refines an initial guess by finding the solution to the initial value problem (for example, by means of trajectory integration) and updating the initial guess until the solution satisfies the boundary conditions.
- Multiple shooting method. The multiple shooting method divides the interval over which the solution is sought into a set of segments and applies the shooting method for each interval. In this case, the method is iterated until the states at the start and end of each segment are matched. As an example, Ref. [24] uses the multiple shooting method to determine Lissajous orbits in the three-body problem.

A simple example

This section presents a simple example of optimization using a linear system with a quadratic cost function of the control variables. The system is general, and it is assumed that the control is continuous. The linear system is given by:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$$

The system is solved using the state transition matrix $\Phi(t, t_0)$. The general solution of the system is now given by:

$$\mathbf{x}(t) = \Phi(t, t_0)\mathbf{x}_0 + \int_{t_0}^t \Phi(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau$$

First, divide the transfer interval into N segments. Assume that the control vector is constant over each segment. The dynamical equations can be written as follows:

$$\mathbf{x}_f = \Phi(t_f, t_0)\mathbf{x}_0 + \sum_{i=1}^N \mathbf{S}_i \mathbf{u}_i, \quad \mathbf{S}_i = \Phi(t_f, t_{i+1}) \int_{t_i}^{t_{i+1}} \Phi(t, \tau)\mathbf{B}(\tau)d\tau$$

The matrices \mathbf{S}_i are found through integration. The dynamical equations can be rewritten by bringing the control inputs to the left-hand side, and the terminal state to the right-hand side.

$$\sum_{i=1}^N \mathbf{S}_i \mathbf{u}_i = \mathbf{x}_f - \Phi(t_f, t_0)\mathbf{x}_0$$

The problem can now been put into a form that is suitable for optimization using the classical method by putting the control variables in a column vector $\mathbf{u} = [\mathbf{u}_1^T \ \dots \ \mathbf{u}_i^T \ \dots \ \mathbf{u}_N^T]^T$. The optimization problem can be stated as follows. Minimize:

$$J = \mathbf{u}^T \mathbf{Q} \mathbf{u}$$

Subject to:

$$\mathbf{A}\mathbf{u} = \mathbf{b}$$

where $\mathbf{A} = [\mathbf{S}_1 \ \dots \ \mathbf{S}_i \ \dots \ \mathbf{S}_N]$ and $\mathbf{b} = \mathbf{x}_f - \Phi(t_f, t_0)\mathbf{x}_0$. The Lagrangian for this problem is:

$$L = \mathbf{u}^T \mathbf{Q} \mathbf{u} + \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{u} - \mathbf{b})$$

To find the optimum, the first derivatives of the Lagrangian are set to zero, and the following set of equations is found:

$$\frac{\partial L}{\partial \mathbf{u}} = \mathbf{Q}\mathbf{u} + \mathbf{A}^T \boldsymbol{\lambda} = 0$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}} = \mathbf{A}\mathbf{u} - \mathbf{b} = 0$$

These equations can be put into matrix form:

$$\begin{bmatrix} \mathbf{Q} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix}$$

The solution for the control vector is found by solving the system of equations through block matrix inversion:

$$\mathbf{u} = \mathbf{Q}^{-1} \mathbf{A}^T (\mathbf{A}\mathbf{Q}^{-1} \mathbf{A}^T)^{-1} \mathbf{b}$$

This solution method produces a discrete approximation to the continuous optimal control input that needs to be applied to drive the system from its initial state \mathbf{x}_0 to its final state \mathbf{x}_f . As mentioned, optimal control is mostly used to generate guidance profiles. Nevertheless, one can appreciate how such formulation is often used in the whole aerospace world. For instance, the reader is suggested to compare the general formulation hereby presented with some important control laws described in [Chapter 10 – Control](#), e.g., linear quadratic regulator.

Interpolation

Interpolation methods are commonly used to represent functions on-board the spacecraft. For example, interpolation methods can be used when optimal trajectories are computed on-ground and control variable histories need to be uploaded to the spacecraft. Another example is the representation of ephemerides of the spacecraft, the Sun, the planets, or other celestial bodies on-board the spacecraft. The objective of interpolation is to reduce the amount of parameter data to be uploaded to the spacecraft.

The subject of interpolation and approximation is vast, and this section is only intended to provide a brief overview of several commonly used methods. In the past, lunar and planetary ephemerides were provided in tabulated format and interpolation and approximation techniques were used extensively to obtain new data points based on the set of tabulated data points. Personal computers (if available) had limited computational power, and it was important to limit the number of calculations to obtain a result. At present, powerful personal computers with internet access are widely available and the relative importance of reducing the number of calculations has decreased. The internet allows rapid dissemination of up-to-date ephemeris data.

Computational resources of satellites tend to be reduced, and at the same time, the ability to transmit and store data is limited. If on-ground calculation of the trajectory is preferred, then the reference trajectory and control inputs need to be uploaded to the spacecraft. It is desirable to ensure that the trajectory is represented in such a way that the amount of data to transmit is as small as possible for a given level of accuracy.

The reference trajectory can be represented in a variety of ways. These representations have in common that they rely on a tabulated set of parameters that form an approximation of the reference trajectory. Desirable characteristics that influence the selection of the type of parameters are the number of parameters required to represent the solution with a certain degree of accuracy and the accuracy over the approximation interval.

Interpolation generally makes use of tabulated data. Classical interpolation methods operate on data presented in tables with the argument in the first column and the function value in the second column. The tables are organized such that the steps of the argument are of equal size. In the classical approach, finite differences of the tabulated values are calculated, as shown in [Table 8.1](#). In this table, the indices of the argument and the function are indicated by subscripted integers. The indices of the uneven differences are multiples of one half, to indicate that these differences are taken

Table 8.1 Tabulated data and finite differences.

Argument	Function	Difference			
		1st	2nd	3rd	4th
t_{-2}	f_{-2}		δ_{-2}^2		
			$\delta_{-\frac{3}{2}}$		$\delta_{-\frac{3}{2}}^3$
t_{-1}	f_{-1}			δ_{-1}^2	δ_{-1}^4
			$\delta_{-\frac{1}{2}}$		$\delta_{-\frac{1}{2}}^3$
t_0	f_0		δ_0^2		δ_0^4
		$\delta_{\frac{1}{2}}$			
t_1	f_1		δ_1^2		δ_1^4
		$\delta_{\frac{3}{2}}$		$\delta_{\frac{3}{2}}^3$	
t_2	f_2		δ_2^2		

between the integer indices, while the even differences have integer values. The order of the differences is indicated by means of a superscript.

Interpolation formulas

The classical methods of interpolation operate on the tabulated function as shown in [Table 8.1](#). The classical methods are polynomial approximations that fit a polynomial through several points in the table. The first derivative is generally not continuous at the start and end of each interval, but the interpolation error is bounded. These methods define the fraction $\tau \in [0, 1]$ of the interpolation interval as follows:

$$\tau_i = \frac{t - t_i}{t_{i+1} - t_i}$$

The interpolation formulas for the classical methods are shown in [Table 8.2](#).

Inverse interpolation

Inverse interpolation is the process of finding the argument for a specific function value. Inverse interpolation can be used to find, for example, the time of eclipse entry or exit or the times at which a ground station becomes visible. Bessel's formula can be used to perform inverse interpolation. The interpolation equation is put into the following form:

$$\tau \delta_{\frac{1}{2}} = f_\tau - f_0 - B^2 (\delta_0^2 + \delta_1^2) - B^3 \delta_{\frac{1}{2}}^3 + B^4 (\delta_0^4 + \delta_1^4) + \dots$$

Table 8.2 Interpolation methods for tabulated data.

Method	Formula
Nearest	$f_\tau = \begin{cases} f_0 & 0 \leq \tau < \frac{1}{2} \\ f_1 & \frac{1}{2} \leq \tau < 1 \end{cases}$
Linear	$f_\tau = f_0 + \tau \delta_{\frac{1}{2}}$
Quadratic	$f_\tau = f_0 + \left(\delta_{\frac{1}{2}} - \frac{1}{2}(1-\tau)\delta_0^2 \right) \tau$
Lagrange second order	$f_\tau = \frac{1}{2}f_{-1}(\tau^2 - \tau) + f_0(1 - \tau^2) + \frac{1}{2}f_1(\tau^2 + \tau)$
Everett	$f_\tau = f_0 + \tau \delta_{\frac{1}{2}} + E_0^2 \delta_0^2 + E_1^2 \delta_1^2 + E_0^4 \delta_0^4 + E_1^4 \delta_1^4 + \dots$ $E_0^2 = -\frac{1}{6}\tau(\tau-1)(\tau-2)$ $E_1^2 = \frac{1}{6}(\tau+1)\tau(\tau-1)$ $E_0^4 = -\frac{1}{120}(\tau+1)\tau(\tau-1)(\tau-2)(\tau-3)$ $E_1^4 = -\frac{1}{120}(\tau+2)(\tau+1)\tau(\tau-1)(\tau-2)$
Bessel	$f_\tau = f_0 + \tau \delta_{\frac{1}{2}} + B^2(\delta_0^2 + \delta_1^2) + B^3 \delta_{\frac{1}{2}}^3 + B^4 (\delta_0^4 + \delta_1^4) + \dots$ $B^2 = \frac{1}{4}\tau(\tau-1)$ $B^3 = \frac{1}{6}\tau(\tau-1)\left(\tau - \frac{1}{2}\right)$ $B^4 = \frac{1}{48}(\tau+1)\tau(\tau-1)(\tau-2)$

In the first step of the iteration, the terms B^2 , B^3 , and B^4 are set to zero to obtain a first approximation of τ . In successive iteration steps, the value of τ is used to compute B^2 , B^3 , and B^4 and refine the estimate until the process converges.

Spline interpolation

Splines are extensively used for interpolation in general to interpolate functions. A spline is a piecewise polynomial that can be used to parameterize curves. Splines tend to have (at least) continuity of the first derivative, but continuity of the second derivative can be enforced. Authors in Ref. [19]

provide an example of how splines can be used in collocation method for trajectory optimization. The basic Hermite spline is defined as follows:

$$f(t) = \begin{cases} C_0(t) & t \in [t_0, t_1] \\ \dots \\ C_i(t) & t \in (t_i, t_{i+1}] \\ \dots \\ C_{n-1}(t) & t \in (t_{n-1}, t_n] \end{cases}$$

In this equation, the piecewise polynomials are third-order polynomial functions of the argument, as follows:

$$C_i(\tau_i) = a_i + b_i \tau_i + c_i \tau_i^2 + d_i \tau_i^3$$

The argument for the interpolation τ is defined in each interval as:

$$\tau_i = \frac{t - t_i}{h_i}, \quad h_i = t_{i+1} - t_i$$

The conditions for each of the segment are as follows. Firstly, the function value at the start and end of the interval needs to be equal to the function to be interpolated:

$$C_i(t_i) = f_i, \quad i = 1, \dots, n - 1$$

$$C_i(t_{i+1}) = f_{i+1}, \quad i = 1, \dots, n - 1$$

The first and second derivatives need to be constant:

$$C'_i(t_{i+1}) = C'_{i+1}(t_{i+1}), \quad i = 1, \dots, n - 2$$

$$C''_i(t_{i+1}) = C''_{i+1}(t_{i+1}), \quad i = 1, \dots, n - 2$$

The second derivative of the spline is continuous and varies linearly over each interval. Fig. 8.4 shows an image of the evolution of the second derivative of the spline over each interval.

The general expression for the polynomial on each segment can be found as:

$$\begin{aligned} C_i(t) = & -f''_i \frac{(t - t_{i+1})^3}{6h_i} + f''_{i+1} \frac{(t - t_i)^3}{6h_i} + \left(f_{i+1} - f''_{i+1} \frac{h_i^2}{6} \right) \frac{t - t_i}{h_i} \\ & - \left(f_i - f''_i \frac{h_i^2}{6} \right) \frac{t - t_{i+1}}{h_i} \end{aligned}$$

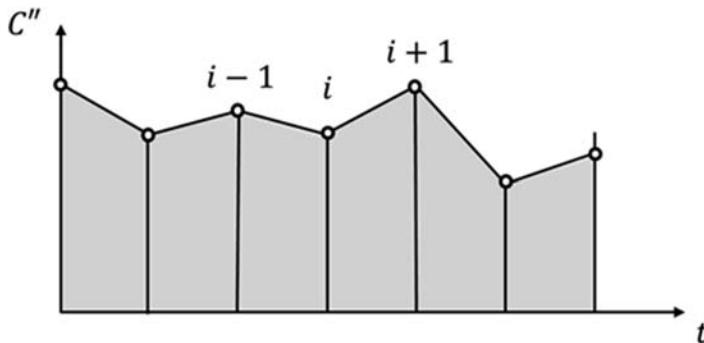


Figure 8.4 Hermite spline interpolation with continuous second derivative.

This equation is generally implemented in matrix form such that all coefficients for the spline are found simultaneously.

Application: rendezvous guidance

The described methods are valid across all the space scenarios. The role of the GNC engineer is to understand and select suitable dynamical models and constraints to generate relevant guidance profiles. The focus of this chapter is the development of guidance functions for formation flying and rendezvous, and the description of the dynamics is briefly revisited with the objective to develop an intuitive understanding of the relative motion and the maneuvers that impact the relative motion. Although the focus is on rendezvous, the same methods can be used for orbit maintenance of a single satellite around its reference position. In addition, the state transition matrix methods and maneuvers that are described below can also be used to calculate trajectory correction maneuvers for interplanetary trajectories.

Relative motion for rendezvous guidance applications

Rendezvous and formation flying depend heavily on orbital mechanics, and the basic solution for propagating the state of a satellite in orbit around a spherical body can be expressed as a composition of functions as follows:

$$\mathbf{x}_1 = \{b(t_1) \circ k(t_1, t_0) \circ b^{-1}(t_0)\}(\mathbf{x}_0)$$

The function \$b\$ expresses the Cartesian coordinates in inertial space as a function of the orbital elements, and the function \$k\$ provides the solution of

Kepler's equation as detailed in [Chapter 4](#) – Orbital Dynamics. This equation can be linearized to study the relative motion around the reference orbit.

$$\delta \mathbf{x}_1 = \mathbf{B}(t_1) \mathbf{K}(t_1, t_0) \mathbf{B}^{-1}(t_0) \delta \mathbf{x}_0$$

The rotation to the local vertical, local horizontal frame can be absorbed into the matrix \mathbf{B} . To understand the dynamics more intuitively, it is useful to start with the relative dynamics around a circular orbit. In the following discussion, the δ indicating infinitesimal changes is dropped for the state vector.

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}$$

The state vector is composed of position and velocity elements and the matrix \mathbf{A} is similarly built up from smaller submatrices. Focusing on the homogeneous part:

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{G} & 2\boldsymbol{\Omega} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix}$$

In the local-vertical-local-horizontal (LVLH) frame (see [Chapter 2](#) – Reference Systems and Planetary Models), the matrix \mathbf{G} contains the accelerations due to the gravity gradient, centrifugal, and Euler force. In a circular orbit, the angular velocity is constant, and the Euler force is equal to zero. The matrix $\boldsymbol{\Omega}$ is the skew-symmetric matrix of the angular velocity of the reference frame. For a circular orbit, the matrix \mathbf{A} is explicitly given by:

$$\mathbf{A}_{i.p.} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2n \\ 0 & 3n^2 & -2n & 0 \end{bmatrix}, \quad \mathbf{A}_{o.p.} = \begin{bmatrix} 0 & 1 \\ -n^2 & 0 \end{bmatrix}$$

The parameter n appearing in the matrix is the orbital rate, given by:

$$n = \sqrt{\frac{\mu}{R^3}} = \frac{2\pi}{T}$$

This is a homogeneous system of equations with constant coefficients. One method to solve the equations of relative motion is to decompose

the matrix \mathbf{A} into the Jordan canonical form. This method provides a direct insight into the nature of the underlying dynamics.

$$\mathbf{J} = \mathbf{P}^{-1} \mathbf{A} \mathbf{P}$$

For a circular orbit, the Jordan decomposition of the in-plane dynamics matrix turns out to be:

$$\begin{aligned} \mathbf{P}_{i.p.} &= \begin{bmatrix} 0 & 1 & 2 & 0 \\ -1 & 0 & 0 & -1 \\ -\frac{3}{2}n & 0 & 0 & -2n \\ 0 & 0 & -n & 0 \end{bmatrix}, \mathbf{P}_{i.p.}^{-1} \\ &= \begin{bmatrix} 0 & -4 & 2n^{-1} & 0 \\ 1 & 0 & 0 & 2n^{-1} \\ 0 & 0 & 0 & -n^{-1} \\ 0 & 3 & -2n^{-1} & 0 \end{bmatrix}, \mathbf{J}_{i.p.} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\frac{3}{2}n & 0 & 0 & 0 \\ 0 & 0 & 0 & -n \\ 0 & 0 & n & 0 \end{bmatrix} \end{aligned}$$

Similarly, the Jordan decomposition of the out-of-plane dynamics matrix is:

$$\mathbf{P}_{o.p.} = \begin{bmatrix} 0 & -1 \\ n & 0 \end{bmatrix}, \mathbf{P}_{o.p.}^{-1} = \begin{bmatrix} 0 & -1 \\ n^{-1} & 0 \end{bmatrix}, \mathbf{J}_{o.p.} = \begin{bmatrix} 0 & n \\ -n & 0 \end{bmatrix}$$

The state transition matrix is found by performing a matrix exponentiation:

$$\Phi(t_1, t_0) = \mathbf{P} e^{\mathbf{J} \Delta t} \mathbf{P}^{-1}$$

The exponentiation of the Jordan matrices leads to the following expressions for the in-plane and out-of-plane motions:

$$\begin{aligned} \mathbf{J}_{i.p.}^{\Delta t} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{3}{2}n\Delta t & 1 & 0 & 0 \\ 0 & 0 & \cos n\Delta t & -\sin n\Delta t \\ 0 & 0 & \sin n\Delta t & \cos n\Delta t \end{bmatrix}, e^{\mathbf{J}_{i.p.} \Delta t} \\ &= \begin{bmatrix} \cos n\Delta t & \sin n\Delta t \\ -\sin n\Delta t & \cos n\Delta t \end{bmatrix} \end{aligned}$$

At this point, the dynamics of rendezvous and formation flying can be understood in simple terms. The first observation is that in the local vertical, local horizontal frame, the in-plane and the out-of-plane motions are uncoupled. The out-of-plane motion has an eigenvalue ni with algebraic multiplicity two (that is, the eigenvalue occurs twice) and geometric multiplicity two (that is, two independent eigenvectors exist). The in-plane motion also has an eigenvalue ni with geometric and algebraic multiplicity, and an eigenvalue 0 with geometric multiplicity two and algebraic multiplicity one, meaning that only one independent eigenvector exists. In the Jordan form, an additional generalized eigenvector is found, along with a coupling factor $\frac{3}{2}n$.

The out-of-plane motion is a simple oscillation, while the in-plane motion is a combination of an oscillation and a drifting motion. The in-plane oscillation manifests itself as a 2×1 ellipse, with initial conditions equal to 1 on the positive z-axis with a velocity $2n$ in the positive x-direction, or initial position 2 on the positive x-axis with a velocity n in the negative z-direction. The drifting motion requires zero velocity for positions on the x-axis. In other words, positions on the x-axis are neutrally stable. Initial conditions with a positive z coordinate equal to 1 require a velocity of $\frac{3}{2}n$ in the positive x-direction. Fig. 8.5 graphically shows the motions associated with the natural modes of the system. The initial conditions associated with the modes are taken from the matrix \mathbf{P} .

Fig. 8.5 identifies the modes as the semimajor axis/mean longitude mode, the eccentricity vector mode, and the inclination vector mode. The Jordan decomposition shows how the in-plane modes are related. First consider a chaser that starts at an altitude of 1. If the velocity in the positive x-direction is equal to $\frac{3}{2}n$, then the semimajor axis mode is excited, but the eccentricity mode is not. On the other hand, if the velocity in the x-direction is equal to $2n$, then the first eccentricity mode is excited, but the semimajor axis mode is not. In addition, when the semimajor axis mode is excited, the mean longitude starts to increase. Next, let's consider a chaser

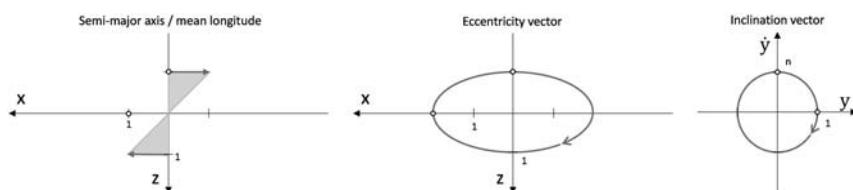


Figure 8.5 In-plane and out-of-plane natural dynamics modes.

located at a position 2 on the x-axis. If the velocity in the negative z-direction is equal to zero, then the mean-longitude mode is excited, and the chaser remains on the x-axis. If the velocity in the negative z-direction is equal to n , then the second eccentricity mode is excited, but the mean longitude mode is not.

To make the link with the orbital elements more explicit, it's possible to decompose the submatrix involving sines and cosines of the transfer time using the angle sum and difference identities for the sine and cosine. This procedure leads to a matrix that only contains sines and cosines of the initial time, and a second matrix that only contains sines and cosines of the terminal time:

$$\begin{bmatrix} \cos n\Delta t & -\sin n\Delta t \\ \sin n\Delta t & \cos n\Delta t \end{bmatrix} = \begin{bmatrix} \cos nt_1 & -\sin nt_1 \\ \sin nt_1 & \cos nt_1 \end{bmatrix} \begin{bmatrix} \cos nt_0 & \sin nt_0 \\ -\sin nt_0 & \cos nt_0 \end{bmatrix}$$

This leads to the following decomposition of the in-plane state transition matrix:

$$\mathbf{B}_{i.p.} = \begin{bmatrix} 0 & 1 & 2 \cos nt_1 & -2 \sin nt_1 \\ -1 & 0 & -\sin nt_1 & -\cos nt_1 \\ -\frac{3}{2}n & 0 & -2n \sin nt_1 & -2n \cos nt_1 \\ 0 & 0 & -n \cos nt_1 & n \sin nt_1 \end{bmatrix},$$

$$\mathbf{B}_{i.p.}^{-1} = \begin{bmatrix} 0 & -4 & 2n^{-1} & 0 \\ 1 & 0 & 0 & 2n^{-1} \\ 0 & 3 \sin nt_0 & -2n^{-1} \sin nt_0 & -n^{-1} \cos nt_0 \\ 0 & 3 \cos nt_0 & -2n^{-1} \cos nt_0 & n^{-1} \sin nt_0 \end{bmatrix},$$

$$\mathbf{K}_{i.p.} = \begin{bmatrix} 1 & -\frac{3}{2}n(t_1 - t_0) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly, the decomposition of the out-of-plane state transition matrix is given by:

$$\mathbf{B}_{o.p.} = \begin{bmatrix} \sin nt_1 & -\cos nt_1 \\ n \cos nt_1 & n \sin nt_1 \end{bmatrix}, \quad \mathbf{B}_{o.p.}^{-1} = \begin{bmatrix} \sin nt_0 & n^{-1} \cos nt_0 \\ -\cos nt_0 & n^{-1} \sin nt_0 \end{bmatrix}, \quad \mathbf{K}_{o.p.}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The in-plane and the out-of-plane matrices \mathbf{K} are now constant, except for the coupling factor $\frac{3}{2}n(t_1 - t_0)$. This indicates that the linear transformation defined by the matrix \mathbf{B} maps a vector of trajectory constants to the relative Cartesian state. The matrix \mathbf{B}^{-1} maps the Cartesian state to a vector of trajectory constants that are related to the orbital elements, cfr. [Chapter 4](#) – Orbital Dynamics. In particular, the rotation by an angle $n\Delta t$ in the matrix $e^{\mathbf{J}\Delta t}$ is replaced by time-dependent transformations in the matrix \mathbf{B} . This implies that the Cartesian modal excitations associated with the eccentricity vector and the inclination vector modes are in fact fixed with respect to inertial space. Small changes in these orbital elements cause the shape or orientation of the orbit to be modified with respect to inertial space. The matrix \mathbf{B} is not unique, just as the Keplerian orbital elements are not the only possible set of trajectory constants. In the current decomposition, the vector of orbital elements differences is labeled as follows: semimajor axis, mean longitude, x component of the eccentricity vector, z component of the eccentricity vector, x component of the inclination vector, and z component of the inclination vector.

$$\delta\boldsymbol{\alpha} = [\delta a \quad \delta\lambda \quad \delta e_x \quad \delta e_z \quad \delta i_x \quad \delta i_z]$$

The dynamics discussed above are a limit case that occurs when eccentricity goes to zero. The family of solutions is continuous in the parameter e , the eccentricity. This means that the insights from this elementary discussion can be applied to the general problem of rendezvous and formation flying around an elliptical orbit. The most straightforward approach to obtain the full solution for the relative motion is to linearize the solution for the Kepler orbit directly (see Refs. [25,26]). Solving the equations of relative motion around an elliptical orbit directly is more challenging, as the dynamics matrix \mathbf{A} becomes a function of time (see Refs. [26,27]).

In the past two decades, a lot of effort has been spent on including perturbations into the state transition matrix, such as, for example, J_2 and other

nonspherical terms, air drag, and solar radiation pressure [28–30]. The solution methods and the solutions found for the state transition matrix are closely related to the development of analytical or semianalytical theories of satellite motions and often introduce additional linear transformations to capture the dynamical effects of the perturbations [30]. Such theories of relative motion by their nature require more computational effort than the simpler theories that only retain the central term of the gravity field.

This understanding of the relative dynamics close to a reference orbit generalizes to an understanding of the behavior of orbits in general. More specifically, the relative dynamics illustrate that spacecraft in lower orbits move faster with respect to higher orbits and vice versa, and that the relative velocity to first order is equal to $\frac{3}{2}n$ times the difference in altitude. Changes in the eccentricity lead to a 2×1 elliptical oscillation around the reference orbit. This geometrical construction, the guiding center approximation, has been used in Ref. [31] to study the near-circular orbits that prevail in the solar system. The out-of-plane oscillations indicate that the orientation of the orbit can be changed by rotating the orbital plane around two perpendicular axes.

The behavior of the relative solutions also provides insight in the evolution of uncertainty. The matrix $\mathbf{K}_{i.p.}$ shows that the semimajor axis and the mean longitude are coupled. This means that if there is uncertainty in the estimation of the semimajor axis, then the uncertainty in the mean longitude will increase over time. The uncertainties of the other elements remain constant in time for this simple model of the dynamics. Fig. 8.6 shows how this works graphically. The uncertainty is represented as an ellipsoid. The initial uncertainty is represented as a gray sphere, which stretches and rotates into

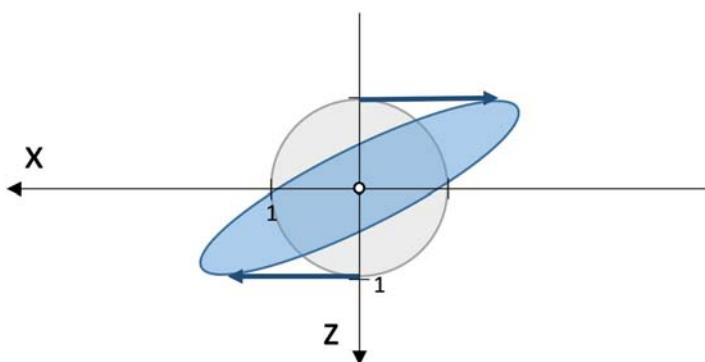


Figure 8.6 Evolution of uncertainty.

an ellipsoid. The combination of rotation and stretching is such that the projection on the z-axis remains unchanged.

Another important observation to make is that the solutions are linear. This observation may seem trivial, but it does allow certain operations and conceptual tools to be applied that cannot be used in nonlinear systems. The most important thing to realize is that relative trajectories can freely be added and subtracted. This can be done to create new relative trajectories, but also to change the origin of the reference frame. In the field of rendezvous, the origin of the LVLH reference frame is usually placed in the center of mass of the target spacecraft. This is a convenient selection because the target spacecraft does not perform any translation maneuvers and therefore remains exactly in the reference orbit. It can be convenient to select a reference orbit that is not linked to any specific spacecraft. For example, in a formation of multiple satellites, the reference orbit can be close to the center of the formation. Relative trajectories between each of the members of the formation can then be examined by means of subtracting the trajectory of the spacecraft that forms the desired origin of the reference frame.

Although the focus of this discussion has been on guidance for rendezvous and formation flying, the state transition matrix for the relative state also has applications to, among others, correction maneuvers for interplanetary trajectories, orbit determination, trajectory optimization, analysis of launch vehicle dispersions, constellation management, collocation and control of geostationary satellites. The linearization around a circular orbit in a corotating frame has been used extensively in celestial mechanics, see Ref. [31].

Effect of velocity impulses

For rendezvous operations, the level of acceleration available from the thrusters is (usually) large enough that desired changes in the velocity can be achieved in a small fraction of the orbital period. This means that the changes in the velocity are essentially impulsive. Impulsive changes in the velocity lead to changes in the evolution of the state vector, and these changes in the evolution of the state vector can be exploited in the design of maneuvers and trajectories for the rendezvous guidance. Fig. 8.7 shows the effect of velocity impulses on the evolution of the in-plane relative dynamics. The application of velocity impulses can be understood by inspecting the inverse matrix $\mathbf{B}_{i.p.}^{-1}$. A tangential impulse directly affects the differential semimajor axis and the eccentricity vector, but it does not

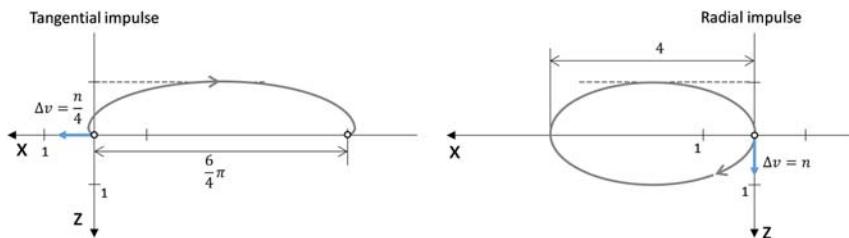


Figure 8.7 Effect of tangential and radial velocity impulses.

directly change the mean longitude. However, the change in semimajor axis immediately starts affecting the mean longitude through the coupling factor $-\frac{3}{2}n$. This leads to a backward drift that is equal to $\frac{6\pi}{n}$ times the ΔV per orbit. A radial impulse directly affects the mean longitude and the eccentricity vector, but it does not change the semimajor axis. The resulting trajectory is a 2×1 ellipse that reaches its farthest point at a distance of $\frac{4}{n}$ times the ΔV after half an orbital period (Table 8.3).

Radial and tangential impulses are the basic building blocks of an in-plane rendezvous guidance, which compares tangential and radial impulses in terms of their effectiveness in changing the dimensions of the relative trajectory. The tangential impulse is more effective overall in the sense that less ΔV is required to affect the change, but the change in the trajectory takes longer to establish. The tangential maneuver also induces along-track drift, which can be beneficial if the objective is to alter the drift rate, but it can also lead to potential risks. An example of such a risk is the inability to perform a maneuver that is aimed to stop the drift due to thruster failure. This drift could in turn lead to collision.

Out-of-plane maneuvers only affect the out-of-plane motion. The out-of-plane motion is a simple harmonic oscillation, and the harmonic oscillation can be controlled by means of impulsive maneuvers occurring at the nodes.

Impulsive maneuvers and trajectories

This section describes a set of maneuvers that can be used to perform rendezvous. The cotangential transfer, the periodic transfer, and the drift modulation transfer generalize the Hohmann transfer, the radial hop, and the tangential hop to make these maneuvers more broadly applicable, while preserving the essential characteristics and objectives of these maneuvers. For these three maneuvers, the simplification to the special cases is presented together with the more general algorithm.

Table 8.3 Comparison of tangential versus radial impulses as means to affect trajectory changes.

Characteristic	Tangential	Radial
Maximum along-track distance	$\frac{6\pi}{n} \Delta V$	$\frac{4}{n} \Delta V$
Time to reach maximum along-track distance	T	$\frac{1}{2} T$
Maximum altitude	$\frac{4}{\eta} \Delta V$	$\frac{1}{\eta} \Delta V$
Time to reach maximum altitude	$\frac{\eta}{2} T$	$\frac{1}{4} T$
Presence of along-track drift	Yes	No

Two-point transfer The two-point transfer maneuver is used to find a transfer trajectory between an initial state vector and a terminal state vector that are given at a fixed initial and final time (which means that the transfer time is fixed). The two-point transfer algorithm is based on inversion of a submatrix of the state transition matrix. Assume a time interval with marked times 0, 1, 2, and 3, where time 0 is the start of the interval, time 3 is the end of the interval, and impulsive maneuvers occur at time 1 and time 2. The evolution of the state vector can then be written as:

$$\mathbf{x}_3 = \Phi_{2 \rightarrow 3} \left(\Phi_{1 \rightarrow 2} \left(\Phi_{0 \rightarrow 1} \mathbf{x}_0 + \begin{bmatrix} \mathbf{0} \\ \Delta \mathbf{v}_1 \end{bmatrix} \right) + \begin{bmatrix} \mathbf{0} \\ \Delta \mathbf{v}_2 \end{bmatrix} \right)$$

This equation is rearranged to the following expression by bringing the impulsive maneuvers to the left-hand side, and the initial and terminal states to the right-hand side and expanding the state transition matrix into position and velocity submatrices.

$$\begin{bmatrix} \Phi_{rr} & \Phi_{rv} \\ \Phi_{vr} & \Phi_{vv} \end{bmatrix}_{1 \rightarrow 2} \begin{bmatrix} \mathbf{0} \\ \Delta \mathbf{v}_1 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \Delta \mathbf{v}_2 \end{bmatrix} = \mathbf{x}_2^*, \quad \mathbf{x}_2^* = \Phi_{2 \rightarrow 3}^{-1} \mathbf{x}_3 - \Phi_{0 \rightarrow 2} \mathbf{x}_0$$

The right-hand side is the state defect at time 2. This equation can be written more compactly by concatenating the impulsive maneuvers into a single vector:

$$\begin{bmatrix} \Phi_{rv,1 \rightarrow 2} & \mathbf{0} \\ \Phi_{vv,1 \rightarrow 2} & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{v}_1 \\ \Delta \mathbf{v}_2 \end{bmatrix} = \mathbf{x}_2^*$$

Inversion of the matrix now leads to the solution for the impulsive maneuvers:

$$\begin{bmatrix} \Delta \mathbf{v}_1 \\ \Delta \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} \Phi_{vv,1 \rightarrow 2}^{-1} & \mathbf{0} \\ -\Phi_{vv,1 \rightarrow 2} \Phi_{vv,1 \rightarrow 2}^{-1} & \mathbf{I}_3 \end{bmatrix} \begin{bmatrix} \mathbf{r}_2^* \\ \mathbf{v}_2^* \end{bmatrix}$$

The matrix $\Phi_{vv,1 \rightarrow 2}$ is:

$$\Phi_{vv,ip} = \begin{bmatrix} -3 + 4 \cos n\Delta t & 2 \sin n\Delta t \\ -2 \sin n\Delta t & \cos n\Delta t \end{bmatrix}, \quad \Phi_{vv,op} = \cos n\Delta t$$

The inverse matrix $\Phi_{vv,1 \rightarrow 2}^{-1}$ for rendezvous in a circular reference orbit is:

$$\Phi_{vv,ip}^{-1} = \frac{n}{\Delta} \begin{bmatrix} \sin n\Delta t & -2 + 2 \cos n\Delta t \\ 2 - 2 \cos n\Delta t & 4 \sin n\Delta t - 3n\Delta t \end{bmatrix}, \quad \Phi_{vv,op}^{-1} = \frac{n}{\sin n\Delta t}$$

The value of the in-plane determinant (scaled by the orbital rate) is:

$$\Delta = 8(1 - \cos n\Delta t) - 3n\Delta t \sin n\Delta t$$

Both the in-plane matrix and the out-of-plane matrix contain a denominator that can become zero. At these points, a singularity occurs in the algorithm. Fig. 8.8 shows a plot of the in-plane determinant as a function of time. The in-plane matrix becomes singular for transfer times that are integer multiples of the orbital period. Another singularity occurs at a value of about 1.406 times the orbital period.

The out-of-plane matrix becomes singular at integer multiples of the orbital period, and at integer multiples, plus one half of an orbital period.

Cotangential (Hohmann) transfer The cotangential transfer is used to modify the altitude of a trajectory using two impulses parallel to V-bar. The cotangential transfer for circular orbits is a modification of the algorithm presented in Ref. [32] for elliptical orbits. The modification is the result of letting the eccentricity go to zero. The basic algorithm is briefly described here. The algorithm uses the following set of transfer parameters:

$$\Delta C_1 = \delta a_2 - \delta a_1, \quad \Delta C_2 = \delta e_{z,2} - \delta e_{z,1}, \quad \Delta C_3 = \delta e_{x,2} - \delta e_{x,1}$$

These transfer parameters define two polynomials that govern the evolution of the z-coordinate in the LVLH frame, and that can be used to determine the cotangential transfer.

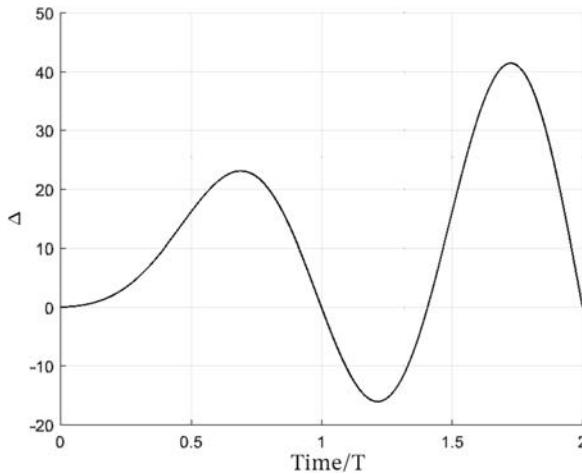


Figure 8.8 Two-point transfer algorithm singularities due to zeros in in-plane determinant Δ .

$$P_1 = \Delta C_1 + \Delta C_2 \cos nt_1 + \Delta C_3 \sin nt_1$$

$$P_2 = \Delta C_2 \sin nt_1 - \Delta C_3 \cos nt_1$$

The polynomials P_1 and P_2 define the sine and the cosine of the transfer angle as follows:

$$\sin n\Delta t = \frac{2P_1 P_2}{P_1^2 + P_2^2}, \quad \cos n\Delta t = \frac{P_2^2 - P_1^2}{P_1^2 + P_2^2}$$

The ΔV 's required for the transfer can be found from the following expressions:

$$\Delta v_{x,1} = -\frac{1}{4} n \frac{(\Delta C_s)^2}{P_1}, \quad \Delta v_{x,2} = \frac{1}{2} n \Delta C_1 - \Delta v_{x,1}$$

The parameter ΔC_s that occurs in the expression for the ΔV is defined based on the following relation:

$$(\Delta C_s)^2 = (\Delta C_2)^2 + (\Delta C_3)^2 - (\Delta C_1)^2$$

The cotangential transfer can be simplified to a Hohmann transfer by assuming that the transfer maneuver only changes the semimajor axis of the orbit and leaves the eccentricity unchanged. In this case, the polynomials governing the transfer simplify to:

$$P_1 = \Delta C_1 = \delta a_2 - \delta a_1, \quad P_2 = 0$$

The sine and the cosine of the transfer angle become 0 and -1 , respectively, indicating that the transfer duration is half an orbital period:

$$\sin n\Delta t = 0, \cos n\Delta t = -1 \Rightarrow n\Delta t = \pi$$

The velocity impulses required at the start and end of the transfer are equal to one quarter times the orbital rate times the difference in semimajor axis, as expected:

$$\Delta v_{x,1} = \frac{1}{4}n(\delta a_2 - \delta a_1), \Delta v_{x,2} = \frac{1}{4}n(\delta a_2 - \delta a_1)$$

The cotangential transfer is a generalization of the Hohmann transfer and can be used as a replacement of the Hohmann transfer in an autonomous on-board guidance. In this situation, the guidance can compute the transfer based on initial conditions that are not ideal for the Hohmann transfer (that is, a lower orbit that is not exactly circular).

Trajectory-crossing maneuver The development of the cotangential transfer maneuver algorithm leads to the observation that this algorithm can become singular when the initial and final trajectories intersect. It is useful to inspect visually the behavior of the z-coordinate and the velocity in the z-direction to understand the crossing maneuver. Fig. 8.9 shows the behavior of the z-coordinate as a function of the parameters that have been identified so far.

The parameter C_m and the angle α that are shown in Fig. 8.9 are defined as follows:

$$(C_m)^2 = (C_2)^2 + (C_3)^2$$

$$\alpha = \tan^{-1}(C_3, C_2)$$

Crossings occur when the radius of the circle defined by C_m is larger than the vertical offset of the circle C_1 . If this is the case, then the locations of the crossings can be found from the following expressions:

$$\sin nt_1 = -\frac{\Delta C_2 \Delta C_s + \Delta C_1 \Delta C_3}{(\Delta C_m)^2}, \quad \cos nt_1 = \frac{\Delta C_3 \Delta C_s - \Delta C_1 \Delta C_2}{(\Delta C_m)^2}$$

$$\sin nt_2 = \frac{\Delta C_2 \Delta C_s - \Delta C_1 \Delta C_3}{(\Delta C_m)^2}, \quad \cos nt_2 = -\frac{\Delta C_3 \Delta C_s + \Delta C_1 \Delta C_2}{(\Delta C_m)^2}$$

The following ΔV 's must be applied at the crossing points.

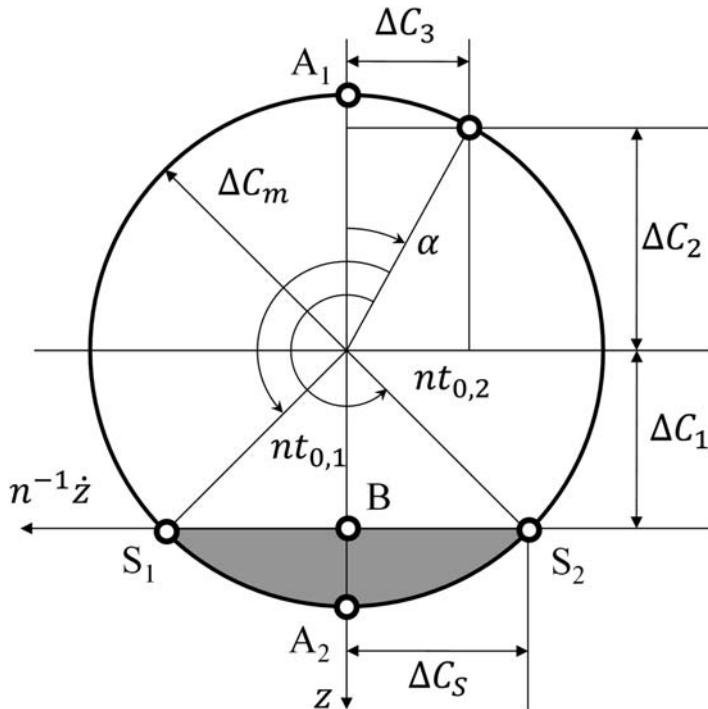


Figure 8.9 Geometry of trajectory crossings. Adapted from T.V. Peters, R. Noomen, Linear cotangential transfers and safe orbits for elliptic orbit rendezvous, *Journal of Guidance, Control, and Dynamics* 44 (2020) (4) 732–748.

$$\Delta \mathbf{v}_{1,ip} = n \begin{bmatrix} \frac{1}{2} \Delta C_1 \\ -\Delta C_s \end{bmatrix}, \quad \Delta \mathbf{v}_{2,ip} = n \begin{bmatrix} \frac{1}{2} \Delta C_1 \\ \Delta C_s \end{bmatrix}$$

The subscripts “1” and “2” indicate the crossing to which each ΔV corresponds. If there is no crossing, it can still be useful to perform a ΔV that cancels the relative drift at the point of closest approach. In Fig. 8.9, this is one of the points at which the z -coordinate reaches its maximum or minimum value. To select the correct point, the following expressions are used.

$$\cos nt = -\text{sgn}(\Delta C_1) \frac{\Delta C_2}{\Delta C_s}, \quad \sin nt = -\text{sgn}(\Delta C_1) \frac{\Delta C_3}{\Delta C_s}$$

The ΔV simply cancels the drift:

$$\Delta \mathbf{v}_{ip} = n \begin{bmatrix} \frac{1}{2} \Delta C_1 \\ 0 \end{bmatrix}$$

The trajectory-crossing maneuver can be used to recompute and replace the second impulse of the cotangential transfer. For an autonomous on-board guidance system, this can be beneficial because it can mitigate the impact of trajectory dispersions that occur as a consequence of navigation and actuation errors and the limited accuracy of the guidance dynamics model.

Periodic (radial hop) transfer The periodic transfer maneuver for circular orbits is based on a similar algorithm presented in Ref. [33] modified in two ways. It has been adapted to circular orbits, and in addition, it has been made possible to include a trajectory drift after the transfer is completed. That is to say, the algorithm makes it possible to transfer from one drifting trajectory to another drifting trajectory by means of a transfer trajectory that is itself drift-free. The tangential components of the first and the second maneuvers are fully determined by the initial and terminal difference in semimajor axis.

$$\Delta v_{x,1} = -\frac{1}{2} n \delta a_1, \quad \Delta v_{x,2} = \frac{1}{2} n \delta a_2$$

The tangential component of the ΔV causes changes in the mean longitude and the components of the eccentricity vector that need to be taken into account in the calculation of the transfer. The matrix $\mathbf{B}_{i,p}^{-1}$ provides the expression for the elements, considering the tangential component of the ΔV .

$$\begin{aligned} \delta \lambda_1^+ &= \delta \lambda_1, \quad \delta e_{x,1}^+ = \delta e_{x,1} - 2 \sin nt_1 n^{-1} \Delta v_{x,1}, \quad \delta e_{z,1}^+ \\ &= \delta e_{z,1} - 2 \cos nt_1 n^{-1} \Delta v_{x,1} \end{aligned}$$

Next, the transfer parameters K_1 , K_2 , and K_3 are formed:

$$K_1 = \frac{1}{2} (\delta \lambda_2 - \delta \lambda_1^+), \quad K_2 = \delta e_{x,2} - \delta e_{x,1}^+, \quad K_3 = -(\delta e_{z,2} - \delta e_{z,1}^+)$$

Transfer polynomials are defined similarly to the cotangential transfer.

$$P_1 = K_1 + K_2 \cos nt_1 + K_3 \sin nt_1$$

$$P_2 = K_2 \sin nt_1 - K_3 \cos nt_1 - \delta a_2$$

The expressions for the sine and cosine of the transfer angle in terms of these polynomials are the same for the nondrifting transfer as for the cotangential transfer.

$$\sin n\Delta t = \frac{2P_1 P_2}{P_1^2 + P_2^2}, \cos n\Delta t = \frac{P_2^2 - P_1^2}{P_1^2 + P_2^2}$$

The radial component of the first and the second ΔV can be found from the following expressions:

$$\Delta v_{z,2} = n \frac{P_1 + \delta a_2 \sin n\Delta t}{1 - \cos n\Delta t}, \Delta v_{z,1} = n K_1 - \Delta v_{z,2}$$

The in-plane ΔV vectors contain the tangential and the radial components of the ΔV provided above:

$$\Delta \mathbf{v}_{1,ip} = \begin{bmatrix} \Delta v_{x,1} \\ \Delta v_{z,1} \end{bmatrix}, \Delta \mathbf{v}_{2,ip} = \begin{bmatrix} \Delta v_{x,2} \\ \Delta v_{z,2} \end{bmatrix}$$

The nondrifting transfer can be simplified to the radial hop. In this case, the differences in semimajor axis at the start and at the end of the trajectory are equal to zero, and the tangential components of the first and second ΔV are both equal to zero. The transfer polynomials are simplified to the following form:

$$P_1 = K_1 = \frac{1}{2}(\delta \lambda_2 - \delta \lambda_1), P_2 = 0$$

This means that the sine and cosine of the transfer angle are 0 and -1 , respectively, meaning that the transfer angle is equal to half an orbital period.

$$\sin n\Delta t = 0, \cos n\Delta t = -1 \Rightarrow n\Delta t = \pi$$

The radial components of the ΔV are equal to one quarter times the orbital rate times the difference in mean anomaly:

$$\Delta v_{z,1} = \frac{1}{4}n(\delta \lambda_2 - \delta \lambda_1), \Delta v_{z,2} = \frac{1}{4}n(\delta \lambda_2 - \delta \lambda_1)$$

The periodic transfer generalizes the radial hop such that it can be used in situations in which the initial conditions are not strictly defined as an along-track displacement. The algorithm can effectively cancel an initial drift, and it can handle initial perturbations of the eccentricity vector by design. Similar to the cotangential transfer, the crossing maneuver can be used as a replacement for the second ΔV . This allows a guidance function to compute a single impulsive maneuver at a time, and it allows the chaser

to mitigate the effects of trajectory errors that may occur because of navigation or actuation errors during the transfer by removing the drift at the end of the transfer.

Drift modulation (tangential hop) transfer Drift modulation alters the along-track distance in a transfer by means of tangential impulses that are spaced in integer multiples of the orbital period. The drift modulation maneuver is a generalization of the tangential hop transfer in the sense that the initial orbital altitude (i.e., semimajor axis) can be nonzero. The first maneuver occurs at time t_1 , and the second maneuver occurs at time $t_1 + NT$. During the N orbital periods of transfer time, the mean-longitude changes by the following amount if no maneuvers are applied:

$$\delta\lambda_f = -3\pi N \delta a_1$$

If the chaser is initially located at a mean longitude $\delta\lambda_1$ and is required to move to a mean longitude $\delta\lambda_2$, then the maneuver needs to perform the following change in mean longitude:

$$\Delta\lambda = \delta\lambda_2 - \delta\lambda_1 - \delta\lambda_f$$

The ΔV 's required for this change can be deduced from Fig. 8.7, and are equal to:

$$\Delta v_{x,1} = -\frac{n}{6\pi N} \Delta\lambda, \quad \Delta v_{x,2} = -\Delta v_{x,1}$$

The drift modulation maneuver can be simplified to the tangential hop by setting the initial difference in semimajor axis to zero. The other equations remain unchanged.

Multiple impulse transfer The multiple-impulse maneuver transfer is a minimum norm solution over the squares of two or more ΔV 's. The derivation of this algorithm is straightforward and can be compared to the simple optimization example in Section A simple example. The terminal state is written as the sum of the contributions of the ΔV 's and the initial state, all propagated to the terminal state using the state transition matrix.

$$\mathbf{x}_f = \sum_{i=1}^{N_{man}} \Phi_{t_{man,i} \rightarrow t_f} \begin{bmatrix} \mathbf{0} \\ \Delta \mathbf{v}_i \end{bmatrix} + \Phi_{t_0 \rightarrow t_f} \mathbf{x}_0$$

These equations can be rewritten as an overdetermined system by rearranging terms:

$$\underbrace{\begin{bmatrix} \Phi_{rv,t_{man,1} \rightarrow t_f} & \dots & \Phi_{rv,t_{man,N} \rightarrow t_f} \\ \Phi_{vv,t_{man,1} \rightarrow t_f} & \dots & \Phi_{vv,t_{man,N} \rightarrow t_f} \end{bmatrix}}_{\mathbf{A}, 3N \times 6} \underbrace{\begin{bmatrix} \Delta \mathbf{v}_1 \\ \vdots \\ \Delta \mathbf{v}_N \end{bmatrix}}_{\mathbf{x}, 3N \times 1} = \underbrace{\mathbf{x}_f - \Phi_{t_0 \rightarrow t_1} \mathbf{x}_0}_{\mathbf{b}, 6 \times 1}$$

The solution for the $\Delta \mathbf{V}$'s is the minimum norm solution, which is related to the least-squares solution for overdetermined systems:

$$\mathbf{x} = \mathbf{A}(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}$$

The solution can be recomputed before each maneuver is applied by updating the system and removing the maneuvers that have already been executed. For the final two maneuvers, the system becomes fully determined, and the solution becomes equal to the two-point transfer solution.

Out-of-plane maneuver The out-of-plane oscillation can be controlled by means of a two-point transfer described in Section [Two-point transfer](#), but it can also be controlled by waiting for the next node to occur and applying a single maneuver there. The out-of-plane motion is parameterized with the components of the inclination vector, and the algorithm uses the differences between the initial and the final desired inclination vector elements.

$$\Delta i_x = \delta i_{x,2} - \delta i_{x,1}, \quad \Delta i_z = \delta i_{z,2} - \delta i_{z,1}$$

The cosine and the sine of the location of the node are found from the following expression.

$$\cos nt_1 = \frac{\Delta i_x}{\Delta i}, \quad \sin nt_1 = \frac{\Delta i_z}{\Delta i}$$

The magnitude of the inclination vector appears in the denominator.

$$\Delta i = \sqrt{(\Delta i_x)^2 + (\Delta i_z)^2}$$

The second node crossing occurs half an orbital period after the first node crossing.

$$t_2 = t_1 + \frac{1}{2} T$$

The ΔV required at the node is equal and opposite to the velocity at the node crossing. More specifically, the following expression provides the ΔV .

$$\Delta v_{y,1} = n(\Delta i_x \cos nt_1 + \Delta i_z \sin nt_1), \quad \Delta v_{y,2} = -\Delta v_{y,1}$$

Forced motion

The strategy for the development of the forced motion guidance is to specify a trajectory and then to compute the feed-forward acceleration required to follow the trajectory.

The forced motion during terminal rendezvous is governed by the same general relative dynamics equations:

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 \\ \mathbf{G} & 2\boldsymbol{\Omega} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_3 \end{bmatrix} \mathbf{u}$$

where \mathbf{u} is the feed-forward control term. This control acceleration is split into an acceleration \mathbf{u}_c compensating for the gravity gradient and fictitious accelerations, and an acceleration \mathbf{u}_k that defines the kinematic accelerations that are required to follow the forced motion trajectory.

$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 \\ \mathbf{G} & 2\boldsymbol{\Omega} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_3 \end{bmatrix} (\mathbf{u}_c + \mathbf{u}_k)$$

The acceleration compensating for the gravity gradient and the fictitious accelerations can be found from the lower part of the dynamics matrix.

$$\mathbf{u}_c = -[\mathbf{G} \quad 2\boldsymbol{\Omega}] \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix}$$

In an ideal setting, the compensating acceleration completely cancels out the gravity gradient and the fictitious accelerations and only the kinematic accelerations required to follow the trajectory remain.

$$\begin{aligned} \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{bmatrix} &= \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 \\ \mathbf{G} & 2\boldsymbol{\Omega} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix} - \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{G} & 2\boldsymbol{\Omega} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_3 \end{bmatrix} \mathbf{u}_k = \\ &= \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_3 \end{bmatrix} \mathbf{u}_k \end{aligned}$$

The compensation for the dynamical effects cannot be expected to be perfect, for several reasons. The compensation depends on the dynamical model, and the guidance model of the dynamics is a simplification of the true dynamics. The compensation depends on the estimated relative state, and this estimated state contains estimation errors. The resulting errors in the feed-forward acceleration provided by the guidance need to be compensated for by the control function.

As an example, consider a forced motion transfer between an initial and a terminal position, with initial and terminal velocities equal to zero.

$$\mathbf{r}(t_0) = \mathbf{r}_0, \mathbf{v}(t_0) = \mathbf{0}, \mathbf{r}(t_1) = \mathbf{r}_1, \mathbf{v}(t_1) = \mathbf{0}$$

A very simple method of performing the forced motion is to assume that the motion starts and ends with an impulsive maneuver, and the velocity during the transfer is constant. In this case, the position and velocity during the transfer are given by:

$$\mathbf{r}(t) = \mathbf{r}_0 + (\mathbf{r}_1 - \mathbf{r}_0) \frac{t}{t_{tf}}, \mathbf{v}(t) = (\mathbf{r}_1 - \mathbf{r}_0) \frac{1}{t_{tf}}$$

The acceleration required to compensate the gravity gradient and the fictitious accelerations is found by evaluating the expression for \mathbf{u}_c for each point of the trajectory.

A more sophisticated example assumes that the chaser performs a simultaneous fly around and radial maneuver. In this case, the position vector is decomposed into a scalar distance and a unit direction vector, both of which are functions of time.

$$\mathbf{r}(t) = r(t)\mathbf{e}_r(t)$$

In this expression, the scalar distance r is the magnitude of the position vector, and the direction vector \mathbf{e}_r points in the direction of the position vector.

$$r = \|\mathbf{r}\|, \mathbf{e}_r = \frac{\mathbf{r}}{\|\mathbf{r}\|}$$

The direction of the initial position vector is taken as the reference direction, and a one-axis rotation is applied to rotate the initial direction vector to the terminal direction vector.

$$\mathbf{e}_r(t) = \mathbf{R}(q(t))\mathbf{e}_0 = q\mathbf{e}_0q^{-1}$$

The rotation quaternion, the angular velocity, and the angular acceleration are determined from the following expression:

$$q(t) = \begin{bmatrix} \mathbf{e}_\alpha \sin \frac{1}{2}\alpha(t) \\ \cos \frac{1}{2}\alpha(t) \end{bmatrix}, \quad \boldsymbol{\omega}(t) = \dot{\alpha}(t)\mathbf{e}_\alpha, \quad \dot{\boldsymbol{\omega}}(t) = \ddot{\alpha}(t)\mathbf{e}_\alpha$$

The rotation angle α_f and rotation axis \mathbf{e}_α can be found from the dot product and the cross product of the initial and the terminal direction vectors:

$$\cos \alpha_f = \mathbf{e}_0 \cdot \mathbf{e}_1, \quad \mathbf{e}_\alpha \sin \alpha_f = \mathbf{e}_0 \times \mathbf{e}_1$$

Now assumptions need to be made on the shape of the functions $r(t)$ and $\alpha(t)$, subject to the following conditions:

$$r(t_0) = r_0, \quad r(t_1) = r_1, \quad \alpha(t_0) = 0, \quad \alpha(t_1) = \alpha_f$$

As in the first example, the functions could simply be linear interpolations between the initial and the final value of the function, but it is also possible to use different types of functions, such as, for example, a cubic Hermite spline. Letting $\tau = \frac{t}{t_{tf}}$:

$$r(t) = (2\tau^3 - 3\tau^2 + 1)r_0 + (-2\tau^3 + 3\tau^2)r_1$$

$$\alpha(t) = (-2\tau^3 + 3\tau^2)\alpha_f$$

The cubic Hermite spline has the advantage that the user can specify the initial and the terminal rate of change of the functions, and in this case, the initial and terminal rates of change are set to zero. This means that there are no discontinuities in velocity at the start and end of the forced motion trajectory, and thus no impulsive ΔV 's are required. The cubic Hermite spline can also be differentiated easily, meaning that the velocity of the reference trajectory and the kinematic feed-forward acceleration required to follow the trajectory can also easily be found. The expression for the velocity and the acceleration along the trajectory is as follows:

$$\dot{\mathbf{r}}(t) = \dot{r}(t)\mathbf{e}_r(t) + r(t)\boldsymbol{\omega}(t) \times \mathbf{e}_r(t)$$

$$\begin{aligned} \ddot{\mathbf{r}}(t) = & \ddot{r}(t)\mathbf{e}_r(t) + 2\dot{r}(t)\boldsymbol{\omega}(t) \times \mathbf{e}_r(t) + r(t)\dot{\boldsymbol{\omega}}(t) \times \mathbf{e}_r(t) \\ & + r(t)\boldsymbol{\omega}(t) \times (\boldsymbol{\omega}(t) \times \mathbf{e}_r(t)) \end{aligned}$$

The acceleration required to compensate the gravity gradient and the fictitious accelerations is again found by evaluating the expression for \mathbf{u}_c for each point of the trajectory.

Application: attitude guidance

Attitude guidance for rendezvous provides reference attitude profiles in terms of the orientation quaternion and the angular velocity. If a feed-forward torque is also requested, Euler's equation can be used to compute this feed-forward torque based on the inertia matrix of the chaser, the angular acceleration, and the angular velocity.

$$\mathbf{t} = \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega})$$

In some cases, it is necessary to compose rotations. For example, the navigation may provide the relative state with respect to the LVLH frame. The attitude guidance may then compute a target-pointing quaternion that provides the required chaser orientation with respect to the LVLH frame. It may be required to provide the chaser orientation with respect to the inertial frame. The composition operation for quaternions, angular velocity, and angular acceleration is given by the following set of equations.

$$\mathbf{Cq}_A = {}_C\mathbf{q}_B \times {}_B\mathbf{q}_A$$

$${}_A\boldsymbol{\omega}_C^C = {}_A\boldsymbol{\omega}_B^C + {}_B\boldsymbol{\omega}_C^C$$

$${}_A\dot{\boldsymbol{\omega}}_C^C = {}_A\dot{\boldsymbol{\omega}}_B^C + {}_A\dot{\boldsymbol{\omega}}_C^C + {}_A\boldsymbol{\omega}_B^C \times {}_A\boldsymbol{\omega}_C^C$$

In this section, a number of basic algorithms are described that are useful for attitude guidance. These algorithms include one- and two-axis pointing, quaternion reorientation, and the LVLH frame orientation with respect to the inertial frame.

One-axis pointing

One-axis pointing can be achieved by forming the shortest-arc quaternion from a reference axis to the desired pointing direction. The one-axis pointing quaternion is formed as:

$$q = \frac{p}{\|p\|}, \quad p = \begin{pmatrix} \mathbf{e}_1 \times \mathbf{e}_2 \\ 1 + \mathbf{e}_1 \cdot \mathbf{e}_2 \end{pmatrix}$$

Quaternions use the sine and cosine of half of the rotation angle instead of the full rotation angle. Fig. 8.10 shows how the one-axis pointing

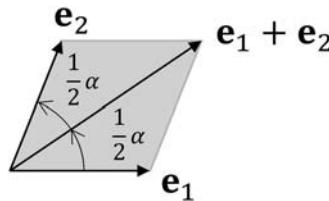


Figure 8.10 Rotation geometry.

quaternion is obtained geometrically. Adding the vector \mathbf{e}_2 to vector \mathbf{e}_1 leads to a vector that is at an angle $\frac{1}{2}\alpha$ with respect to vector \mathbf{e}_1 .

The nonnormalized quaternion p is formed using the dot product and the cross product of the vectors \mathbf{e}_1 and $\mathbf{e}_1 + \mathbf{e}_2$.

$$\begin{aligned} p &= \begin{pmatrix} \mathbf{e}_1 \times (\mathbf{e}_1 + \mathbf{e}_2) \\ \mathbf{e}_1 \cdot (\mathbf{e}_1 + \mathbf{e}_2) \end{pmatrix} = \begin{pmatrix} \mathbf{e}_1 \times \mathbf{e}_1 + \mathbf{e}_1 \times \mathbf{e}_2 \\ \mathbf{e}_1 \cdot \mathbf{e}_1 + \mathbf{e}_1 \cdot \mathbf{e}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{e}_1 \times \mathbf{e}_2 \\ 1 + \mathbf{e}_1 \cdot \mathbf{e}_2 \end{pmatrix} \\ &= \begin{pmatrix} \sin \alpha \mathbf{e}_3 \\ 1 + \cos \alpha \end{pmatrix} \end{aligned}$$

The unit vector \mathbf{e}_3 is perpendicular to both vector \mathbf{e}_1 and \mathbf{e}_2 . The norm of this quaternion is equal to:

$$\|p\| = \sqrt{2(1 + \cos \alpha)}$$

The normalized quaternion turns out to contain the sine and cosine of half the rotation angle, as expected.

$$\begin{aligned} q &= \frac{1}{\sqrt{2(1 + \cos \alpha)}} \begin{pmatrix} \sqrt{1 - \cos^2 \alpha} \mathbf{e}_3 \\ 1 + \cos \alpha \end{pmatrix} \\ &= \frac{1}{\sqrt{2}} \begin{pmatrix} \sqrt{1 - \cos \alpha} \mathbf{e}_3 \\ \sqrt{1 + \cos \alpha} \end{pmatrix} = \begin{pmatrix} \sin \frac{1}{2} \alpha \mathbf{e}_3 \\ \cos \frac{1}{2} \alpha \end{pmatrix} \end{aligned}$$

The angular velocity associated with the quaternion is found from the quaternion derivative and the quaternion conjugate, namely:

$$\boldsymbol{\omega} = 2\dot{q}q^*$$

The quaternion q is obtained by normalizing the quaternion p , such that the quaternion derivative of q can be found:

$$\dot{q} = \frac{1}{\|p\|} \frac{d}{dt} p + p \frac{d}{dt} \frac{1}{\|p\|} = \frac{\dot{p}}{\|p\|} - \left(\frac{p}{\|p\|} \cdot \frac{\dot{p}}{\|p\|} \right) \frac{p}{\|p\|}$$

The derivative of the quaternion p is found from the derivatives of the unit vectors on which it is based:

$$\dot{p} = \begin{bmatrix} \dot{\mathbf{e}}_1 \times \mathbf{e}_2 + \mathbf{e}_1 \times \dot{\mathbf{e}}_2 \\ \dot{\mathbf{e}}_1 \cdot \mathbf{e}_2 + \mathbf{e}_1 \cdot \dot{\mathbf{e}}_2 \end{bmatrix}$$

Similarly, the angular acceleration vector can be found by differentiating the expression for the angular velocity.

$$\dot{\boldsymbol{\omega}} = 2\ddot{q}\dot{q}^* + 2\dot{q}\dot{q}^*$$

The derivative of the quaternion conjugate is found from the expression:

$$\dot{q}^* = -q^* \dot{q} q^*$$

The second derivative of the normalized quaternion q can be found from the nonnormalized quaternion as follows.

$$\ddot{q} = \frac{\ddot{p}}{\|p\|} - 2 \left(\frac{p}{\|p\|} \cdot \frac{\dot{p}}{\|p\|} \right) \frac{\dot{p}}{\|p\|} - \left\{ \left(\frac{\dot{p}}{\|p\|} \cdot \frac{\dot{p}}{\|p\|} \right) - 3 \left(\frac{p}{\|p\|} \cdot \frac{\dot{p}}{\|p\|} \right) \right. \\ \left. \times \left(\frac{p}{\|p\|} \cdot \frac{\dot{p}}{\|p\|} \right) + \left(\frac{p}{\|p\|} \cdot \frac{\ddot{p}}{\|p\|} \right) \right\} \frac{p}{\|p\|}$$

Finally, the second derivative of the nonnormalized quaternion p is found from the unit vectors and their derivatives.

$$\ddot{p} = \begin{bmatrix} \ddot{\mathbf{e}}_1 \times \mathbf{e}_2 + 2\dot{\mathbf{e}}_1 \times \dot{\mathbf{e}}_2 + \mathbf{e}_1 \times \ddot{\mathbf{e}}_2 \\ \ddot{\mathbf{e}}_1 \cdot \mathbf{e}_2 + 2\dot{\mathbf{e}}_1 \cdot \dot{\mathbf{e}}_2 + \mathbf{e}_1 \cdot \ddot{\mathbf{e}}_2 \end{bmatrix}$$

As an example, consider the quaternion that rotates the x-axis from the base frame to a position vector \mathbf{r} . In this case:

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{e}_2 = \frac{\mathbf{r}}{\|\mathbf{r}\|}$$

$$\dot{\mathbf{e}}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \dot{\mathbf{e}}_2 = \frac{\mathbf{v}}{\|\mathbf{r}\|} - \left(\mathbf{e}_2 \cdot \frac{\mathbf{v}}{\|\mathbf{r}\|} \right) \mathbf{e}_2$$

$$\ddot{\mathbf{e}}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \ddot{\mathbf{e}}_2 = \frac{1}{\|\mathbf{r}\|} \left(\mathbf{a} - 2(\mathbf{v} \cdot \mathbf{e}_2) \dot{\mathbf{e}}_2 - (\mathbf{a} \cdot \mathbf{e}_2 + \mathbf{v} \cdot \dot{\mathbf{e}}_2) \mathbf{e}_2 \right)$$

Two-axis pointing

Two-axis pointing is useful when a spacecraft needs to point the principal axis in a certain direction while minimizing the angle between a secondary axis and a specific secondary direction. For example, a chaser spacecraft may need to point a camera in the direction of a point of interest while minimizing the angle between the solar panel normal and the Sun direction.

The algorithm makes use of an extended vector normalization algorithm that provides the first and second rate of change of the unit vector that are used to obtain the angular velocity and the angular acceleration.

The algorithm takes the first pointing vector \mathbf{r}_1 and the secondary pointing vector \mathbf{r}_2 and forms the cross product. The vector \mathbf{r}_3 is perpendicular to both \mathbf{r}_1 and \mathbf{r}_2 . The velocity and acceleration vectors associated with the first and the second pointing vectors are used to find the first and second rate of change of the vector \mathbf{r}_3 .

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$$

$$\mathbf{v}_3 = \mathbf{v}_1 \times \mathbf{r}_2 + \mathbf{r}_1 \times \mathbf{v}_2$$

$$\mathbf{a}_3 = \mathbf{a}_1 \times \mathbf{r}_2 + 2\mathbf{v}_1 \times \mathbf{v}_2 + \mathbf{r}_1 \times \mathbf{a}_2$$

The extended normalization algorithm is used to find the unit vector \mathbf{e}_1 and its derivatives $\dot{\mathbf{e}}_1$ and $\ddot{\mathbf{e}}_1$, and the unit vector \mathbf{e}_3 and its derivatives $\dot{\mathbf{e}}_3$ and $\ddot{\mathbf{e}}_3$. The unit vector \mathbf{e}_x is identified with the vector \mathbf{e}_1 , while the unit vector \mathbf{e}_z is identified with the vector \mathbf{e}_3 . The unit vector \mathbf{e}_y is obtained from:

$$\mathbf{e}_y = \mathbf{e}_z \times \mathbf{e}_x$$

$$\dot{\mathbf{e}}_y = \dot{\mathbf{e}}_z \times \mathbf{e}_x + \mathbf{e}_z \times \dot{\mathbf{e}}_x$$

$$\ddot{\mathbf{e}}_y = \ddot{\mathbf{e}}_z \times \mathbf{e}_x + 2\dot{\mathbf{e}}_z \times \dot{\mathbf{e}}_x + \mathbf{e}_z \times \ddot{\mathbf{e}}_x$$

The angular velocity vector is found from the following expressions:

$$\omega_x = \frac{1}{2} (\dot{\mathbf{e}}_y \cdot \mathbf{e}_z - \mathbf{e}_y \dot{\mathbf{e}}_z)$$

$$\omega_y = \frac{1}{2} (-\dot{\mathbf{e}}_x \cdot \mathbf{e}_z + \mathbf{e}_x \dot{\mathbf{e}}_z)$$

$$\omega_z = \frac{1}{2} (\dot{\mathbf{e}}_x \cdot \mathbf{e}_y - \mathbf{e}_x \dot{\mathbf{e}}_y)$$

The angular acceleration vector is found from similar expressions:

$$\dot{\omega}_x = \frac{1}{2} (\ddot{\mathbf{e}}_y \cdot \mathbf{e}_z - \mathbf{e}_y \ddot{\mathbf{e}}_z)$$

$$\dot{\omega}_y = \frac{1}{2} (-\ddot{\mathbf{e}}_x \cdot \mathbf{e}_z + \mathbf{e}_x \ddot{\mathbf{e}}_z)$$

$$\dot{\omega}_z = \frac{1}{2} (\ddot{\mathbf{e}}_x \cdot \mathbf{e}_y - \mathbf{e}_x \ddot{\mathbf{e}}_y)$$

Extended vector normalization The extended vector normalization computes the unit vector, as well as the rate of change and the second rate of change of an input vector \mathbf{v} , and its first and second derivatives $\dot{\mathbf{v}}$ and $\ddot{\mathbf{v}}$. First, the input vector is normalized:

$$\mathbf{e}_v = \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

Next, the auxiliary parameters $\dot{\mathbf{u}}$ and $\ddot{\mathbf{u}}$ are formed:

$$\dot{\mathbf{u}} = \frac{\dot{\mathbf{v}}}{\|\mathbf{v}\|}, \quad \ddot{\mathbf{u}} = \frac{\ddot{\mathbf{v}}}{\|\mathbf{v}\|},$$

Using these expressions, the first and second rate of change of the unit vector are formed by removing the component of the derivative vectors that lie along the unit vector:

$$\dot{\mathbf{e}}_v = \dot{\mathbf{u}} - (\dot{\mathbf{u}} \cdot \mathbf{e}_v) \mathbf{e}_v$$

$$\ddot{\mathbf{e}}_v = \ddot{\mathbf{u}} - 2(\dot{\mathbf{u}} \cdot \mathbf{e}_v) \dot{\mathbf{e}}_v - (\ddot{\mathbf{u}} \cdot \mathbf{e}_v + \dot{\mathbf{u}} \cdot \dot{\mathbf{e}}_v) \mathbf{e}_v$$

Reorientation

For the reorientation guidance, the algorithm in Ref. [34] can be used. This algorithm provides an interpolation between an initial attitude state (consisting of a quaternion and an angular velocity) to a terminal attitude state performed in a given amount of time. For an effective reorientation, the difference between the initial and the desired quaternion is formed; this is the error quaternion. The error quaternion is interpolated from the initial value to zero within the desired reorientation time. At each time step of the transfer, the attitude guidance computes the reference attitude as the desired quaternion multiplied by the error quaternion. In doing so, the reference attitude trajectory is equal to the initial orientation at the start of the transfer and equal to the final attitude at the end of the transfer.

Four interpolation coefficients q_4 , ω_1 , ω_2 , and ω_3 are generated as follows:

$$q_4 = q_a$$

$$\omega_1 = \frac{1}{3}\omega_a t_{tf}$$

$$\omega_2 = \log\left(\left(\exp\left(\frac{1}{3}\omega_a t_{tf}\right)\right)^* q_a^* q_b \left(\exp\left(\frac{1}{3}\omega_b t_{tf}\right)\right)^*\right)$$

$$\omega_3 = \frac{1}{3}\omega_b t_{tf}$$

The superscript * indicates the quaternion conjugate. The quaternion exponent and logarithm are defined as follows:

$$\exp(\mathbf{v}) = q_v = \begin{pmatrix} \mathbf{e}_v \sin \vartheta \\ \cos \vartheta \end{pmatrix}, \mathbf{e}_v = \frac{\mathbf{v}}{|\mathbf{v}|}, \vartheta = |\mathbf{v}|$$

$$\log(q_v) = \mathbf{v}$$

The quaternion is given by a scalar component and a vector component, cfr. [Chapter 5](#) – Attitude Dynamics:

$$\mathbf{q} = \left\{ \begin{array}{l} \mathbf{q}_{1:3} \\ q_4 \end{array} \right\}$$

The quaternion logarithm outputs the following vector:

$$\mathbf{v} = \vartheta \mathbf{e}_{\mathbf{q}}, \quad \mathbf{e}_{\mathbf{q}} = \frac{\mathbf{q}}{|\mathbf{q}|}, \quad \vartheta = 2 \tan^{-1}(|\mathbf{q}_{1:3}|, q_4)$$

The following interpolating polynomials are defined:

$$\beta_1 = 1 - (1 - \tau)^3$$

$$\beta_2 = 3\tau^2 - 2\tau^3$$

$$\beta_3 = \tau^3$$

The interpolated quaternion is given by the following function of the four interpolation coefficients q_4 , ω_1 , ω_2 , and ω_3 :

$$q = q_4 \exp(\omega_1 \beta_1) \exp(\omega_2 \beta_2) \exp(\omega_3 \beta_3)$$

The first and second derivatives of these expressions are used to construct the angular velocity and angular acceleration. The feed-forward torque is computed using the already presented equation $\mathbf{t} = \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega})$.

Quaternion rotation: LVLH, PQW, and RSW

The rotation from the inertial frame to the LVLH frame is composed by a sequence of rotations. The relationship between the Earth-centered inertial (ECI) frame and the LVLH frame has been presented in [Chapter 2 – Reference systems](#). Here, we present a quaternion perspective on the rotation, which is useful for the generation of guidance profiles. The first set of rotations rotates the inertial frame to the LVLH frame at perigee. The LVLH frame used in this first rotation is the comoving frame LVLH-based $\mathbf{\Gamma}_{m,n,p}$, in which, differently from $\Delta_{i,j,k}$, the radial direction toward the attracting body is called z axis, whereas the y axis is opposite to the angular momentum vector (please refer to [Chapter 2 – Reference systems](#)). The first three rotations through the angles Ω , i , and ω is the familiar orbit rotation sequence [11]. The two rotations through the angle $\frac{\pi}{2}$ aim to make the y -axis point toward the negative orbital angular momentum vector and the z -axis of the frame toward the center of the Earth.

$${}_{\pi}\mathbf{A}_I = \mathbf{R}_z\left(\frac{\pi}{2}\right) \mathbf{R}_y\left(-\frac{\pi}{2}\right) \mathbf{R}_z(\omega) \mathbf{R}_x(i) \mathbf{R}_z(\Omega)$$

The rotation to the *LVLH* frame is obtained by performing a final rotation through the true anomaly.

$${}_{LVLH}\mathbf{A}_I = \mathbf{R}_y(-\vartheta) \mathbf{R}_{I \rightarrow \pi}$$

This rotation sequence can be written as a quaternion as follows, where again the first three components are the vectorial part as $\mathbf{q} = \begin{Bmatrix} \mathbf{q}_{1:3} \\ q_4 \end{Bmatrix}$.

$$_{LVLH}\mathbf{q}_I = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

The elements of this quaternion are:

$$q_1 = -\cos \frac{i}{2} \cos \frac{1}{2} (\Omega + \omega + \vartheta) + \cos \frac{i}{2} \sin \frac{1}{2} (\Omega + \omega + \vartheta)$$

$$+ \sin \frac{i}{2} \cos \frac{1}{2} (\Omega - \omega - \vartheta) + \sin \frac{i}{2} \sin \frac{1}{2} (\Omega - \omega - \vartheta)$$

$$q_2 = -\cos \frac{i}{2} \cos \frac{1}{2} (\Omega + \omega + \vartheta) - \cos \frac{i}{2} \sin \frac{1}{2} (\Omega + \omega + \vartheta)$$

$$- \sin \frac{i}{2} \cos \frac{1}{2} (\Omega - \omega - \vartheta) + \sin \frac{i}{2} \sin \frac{1}{2} (\Omega - \omega - \vartheta)$$

$$q_3 = \cos \frac{i}{2} \cos \frac{1}{2} (\Omega + \omega + \vartheta) + \cos \frac{i}{2} \sin \frac{1}{2} (\Omega + \omega + \vartheta)$$

$$- \sin \frac{i}{2} \cos \frac{1}{2} (\Omega - \omega - \vartheta) + \sin \frac{i}{2} \sin \frac{1}{2} (\Omega - \omega - \vartheta)$$

$$q_4 = \cos \frac{i}{2} \cos \frac{1}{2} (\Omega + \omega + \vartheta) - \cos \frac{i}{2} \sin \frac{1}{2} (\Omega + \omega + \vartheta)$$

$$+ \sin \frac{i}{2} \cos \frac{1}{2} (\Omega - \omega - \vartheta) + \sin \frac{i}{2} \sin \frac{1}{2} (\Omega - \omega - \vartheta)$$

The two rotations through the angle $\frac{\pi}{2}$ complicate the expression for the quaternion considerably. The quaternion that encodes the rotation from the inertial frame to the perifocal frame is:

$$_{PQW}\mathbf{q}_I = \begin{bmatrix} \sin \frac{i}{2} \cos \frac{1}{2} (\Omega - \omega) \\ \sin \frac{i}{2} \sin \frac{1}{2} (\Omega - \omega) \\ \cos \frac{i}{2} \sin \frac{1}{2} (\Omega + \omega) \\ \cos \frac{i}{2} \cos \frac{1}{2} (\Omega + \omega) \end{bmatrix}$$

The rotation from the inertial frame to the radial, tangential normal (RSW) frame $\Delta_{i,j,k}$ (please refer to [Chapter 2 – Reference systems](#)) is given by:

$$RSW\boldsymbol{q}_I = \begin{bmatrix} \sin \frac{i}{2} \cos \frac{1}{2}(\Omega - \omega - \vartheta) \\ \sin \frac{i}{2} \sin \frac{1}{2}(\Omega - \omega - \vartheta) \\ \cos \frac{i}{2} \sin \frac{1}{2}(\Omega + \omega + \vartheta) \\ \cos \frac{i}{2} \cos \frac{1}{2}(\Omega + \omega + \vartheta) \end{bmatrix}$$

In the RSW frame, the x-axis points radially outward, the z-axis points in the direction of the orbital angular momentum vector, and the y-axis completes the right-handed frame.

Design of a guidance function

This section explores the design of a guidance function at system level. It covers the generation and derivation of requirements to the architecture definition.

Identification of guidance requirements

The design of the guidance is an iterative process that typically begins with the identification of requirements for the guidance. The requirements specify the organization, functionalities, and the performance of the guidance function. The set of requirements serves as a starting point for designing and implementing the guidance function. The requirements for the guidance can be grouped into sets that cover the following domains:

- Functional
 - Division into functional elements.
 - Specification of functions to be performed by the guidance.
 - Specification of modes contained in the guidance.
 - Constraints to be taken into account by the guidance.
 - Specification of types of maneuvers to be considered.
 - Safety.
- Performance
 - Safety.
 - Propellant or ΔV consumption.
 - Margins with respect to available actuator forces and torques.
 - CPU load.
 - Accuracy.

The functional requirements define the capabilities of the guidance. Functional requirements divide the guidance into functional elements. For rendezvous guidance, such elements could be, for example, impulsive guidance, forced motion guidance, and attitude guidance, safety monitoring, and others. The functional requirements may also specify the functions that the guidance needs to perform. For rendezvous, this could be, for example, performing a sequence of maneuvers to approach the target, perform target pointing, etc. The functional requirements may specify the modes that need to be present within the guidance. The guidance modes are discussed more in detail in Section [Guidance modes](#). The functional requirements may also specify constraints to be taken into account by the guidance, such as not pointing instruments or radiator panels toward the Sun, or pointing Solar panels toward the Sun. Another aspect that may be covered by this kind of requirements is the type of maneuvers that need to be considered. This could include, for example, collision avoidance maneuvers or maneuvers to enter a passively safe formation. The functional requirements can also cover certain aspects of safety. For rendezvous, this may include, for example, requirements on using passively safe trajectories during the approach or requirements that state that the trajectory needs to remain safe even if any of the impulsive ΔV 's cannot be applied.

Performance requirements, as the name implies, define the performance of the guidance in terms of numerical values. Safety performance requirements could, for example, require a safe relative trajectory to remain collision-free for a certain number of days, or it could specify a margin around the nominal trajectory that the chaser cannot leave. Requirements can also be set on propellant consumption and the magnitude of maneuvers. For example, the long-range rendezvous may be required to consume less than a certain amount of propellant. Performance requirements may also specify the margins that need to be respected with respect to the maximum forces and torques available from the actuators in order to leave sufficient margin for the control. For example, feed-forward torque may be required to be less than 70% of the total torque available from the reaction wheels. The performance requirements may also specify the central processing unit (CPU) load that the guidance function cannot exceed, for example, 10% of the total maximum CPU load. Such requirements may require tests to be performed on representative hardware; in this case, running the software on a space-qualified CPU. Accuracy requirements can be related to safety, for example, if a requirement states that the chaser at arrival after a two ΔV transfer needs to have an accuracy, that is better than a certain

percentage of the distance of the arrival point to the target. This would allow the design of the guidance trajectory to become robust to uncertainty in the navigation solution when the transfer is calculated, to uncertainties in the dynamical model used, and to errors in the maneuver execution. Another example could be the accuracy with which Sun pointing is achieved from the guidance perspective. In this case, a functional requirement may specify that a Sun ephemeris is present in the guidance, and a performance requirement may specify the required angular accuracy of the ephemeris. These performance requirements are closely related to the creation of (pointing) error budgets and the allocation of allowable maximum errors to each of the components of the error budget. Reference [2] contains an extensive discussion on how to create pointing error budgets and specifically how the process of creating the error budget is related to modeling and analyzing the system as a whole. The process of creating an error budget is divided into the following steps:

1. Modeling the system and identifying all error sources.
2. Characterization of each of the error sources (magnitude, type of distribution).
3. Combining all error sources to form the overall system error.

The system modeling and the identification of the error sources includes a determination of the sensitivity of the overall system error to each of the error components. The combination of all error sources into an overall system error can be a challenging task, and Monte Carlo simulations or worst-case analyses may be required to determine whether the performance of the system as a whole is sufficient. For a rendezvous mission, the performance accuracy may be determined by the overall error at docking. A Monte Carlo test campaign may need to be run that varies parameters such as sensor and actuator misalignments, uncertainties in the level of thrust available for each thruster, uncertainties in the mass properties of the chaser, uncertainties in the level of perturbations (e.g., vibration frequencies of the solar panels, propellant slosh), and many others. Different performance metrics may apply to different phases of the mission. From the guidance perspective, the main impact that the guidance can have is in the accuracy of the models that are used to perform the calculations. Secondary impacts of the guidance on the performance accuracy are the smoothness of the profiles provided by the guidance and the provision of feed-forward terms. Returning to the example of the Sun-pointing accuracy, it can be observed that the Sun-pointing accuracy depends on the accuracy of the orbit determination, the accuracy of the on-board clock, the overall attitude pointing accuracy,

which may depend on as well as the accuracy of the ephemeris model. If the guidance includes the ephemeris model, then the guidance can only influence the Sun-pointing accuracy through the accuracy of the ephemeris model. From the point of view of the guidance, it makes sense to adjust the accuracy of the ephemeris to be in line with the errors expected for the other components of the error budget.

Finally, the requirements need to indicate how the requirements are verified. Requirements can be verified in the following ways:

- Test.
- Inspection.
- Analysis.
- Review of design.

The most useful method of verification is through testing. Testing of the guidance can be performed at the level of single utility functions, modes, the entire guidance function, and the guidance function as integrated within the overall GNC software. Performance requirements are generally testable. Inspection of the design can verify whether certain functionalities are present such as, for example, the ability to go to a safe configuration. Analysis as a method of verification can be useful if direct testing is not practical. For example, if a requirement specifies long-term safety after a sequence of maneuvers, then it may be impractical to run a GNC simulator (which operates at the frequency of the GNC) for the duration of the desired interval in a Monte Carlo campaign, because this would require too much computation time. It may be more convenient to determine the terminal conditions after the sequence of maneuvers and to use a stand-alone orbit propagator to analyze the safety. Finally, a review of the design (usually in the form of written documentation) can be used to verify certain types of requirements. It may, for example, be the case that a functional requirement states that the guidance needs to calculate the ephemeris of the Sun; this can be verified by means of checking the design documentation.

Guidance modes

A mode is a configuration of the (sub)system in a certain state that is characterized by the active functions or enabled functional capabilities. The mode of the system specifies the behavior of the system or subsystem. For rendezvous and formation flying, the guidance may need to perform different functions during different parts of the mission. Fig. 8.11 shows how the guidance function for rendezvous can be divided into different domains (translation, attitude, and special functions) and into different modes.

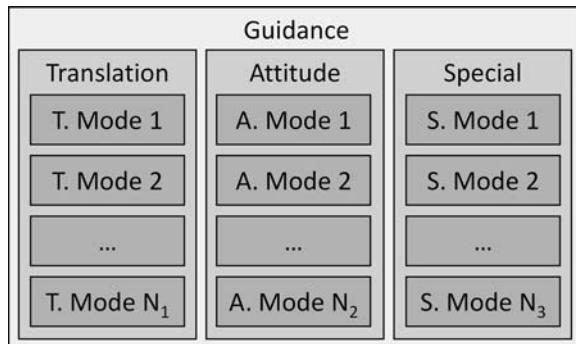


Figure 8.11 Guidance function divided into translation, attitude, and special guidance functions, each with distinct modes.

Each of these modes has a different objective, and different combinations of translation, attitude, and guidance modes can be available during different phase of the mission. For example, the translation guidance can provide a set of impulsive maneuvers during a mode that is aimed at maintaining a specific trajectory, while the attitude guidance may switch between target pointing mode and ΔV pointing mode as the maneuvers need to be executed.

The different guidance modes give the guidance the capability to have different behaviors that apply to different types of situations. This flexibility to adapt its behavior to the environment and to the current objectives enable the overall GNC system to operate with a higher degree of autonomy.

From the perspective of design, development, validation, and verification, it is also advantageous to break down the guidance into distinct functional modes because each of the modes represents a smaller, simpler functional unit that can be tested independently from the other modes.

The simple division into modes that is shown in Fig. 8.11 does not indicate very well how data and commands flow through the guidance model. The architecture of the guidance provides a high-level view of the way the guidance operates and is the subject of the next section.

Architecture

Fig. 8.12 shows a generic design for the guidance architecture. A guidance mode manager manages the lower-level guidance modes, responding to the GNC mode manager. Pre- and postprocessing blocks are present to perform calculations that are applicable to all the modes. For example, assume that all modes require the current orientation of the chaser with respect to the

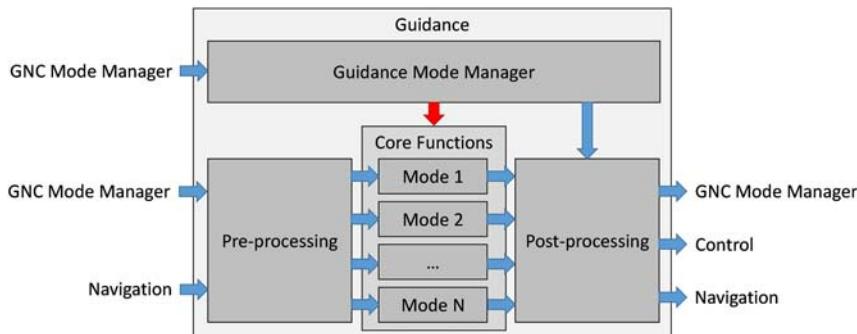


Figure 8.12 Guidance architecture indicating internal flow of data.

LVLH frame. Also, assume that the input to the guidance provides the orientation with respect to the inertial frame and the orientation of the LVLH frame with respect to the inertial frame. The preprocessing function would then contain the functions required to compute the orientation of the chaser with respect to the LVLH frame from the input. The preprocessing function also ensures that the correct information is available to each of the modes, and the postprocessing ensures that the output of the guidance is padded if necessary. It is possible, for example, that not all modes require exactly the same input or produce the same output. In this case, the preprocessing ensures that the correct input data are provided to each of the modes. The postprocessing block can pass the outputs that are not provided by specific modes, if required. For example, the impulsive translational guidance does not provide feed-forward accelerations, while the forced motion guidance does. The postprocessing function ensures that the output of the overall guidance function contains all the possible outputs that the modes can provide. When the impulsive guidance modes are active, the postprocessing function may fill the feed-forward acceleration with zeroes.

The postprocessing function may also contain functions that apply to some of the modes, but not to all. In such cases, some freedom exists in determining where best to place the respective functions. For example, free-flying trajectory propagation for the generation of reference trajectories may apply to the impulsive guidance modes, but not to forced motion guidance modes. In this case, the trajectory propagation could be incorporated inside each of the modes, or a free-flying trajectory propagation can be incorporated inside the postprocessing with an activation switch that ensures that it is only active during the impulsive maneuvering modes. This example should illustrate that the guidance architectural design is a useful conceptual

tool, but that on occasion, practical design considerations should ensure that the most convenient organization of the functions of the guidance is chosen. As such, the guidance architectural design should help to make the guidance as easy to understand as possible.

Function library

The algorithms described in Section [Design process](#) can be implemented into function libraries that can in turn be used to construct the guidance. Taking a generic impulsive maneuver calculation as an example, a generic maneuver calculation command could look as follows:

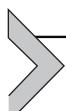
$$[t, \Delta\mathbf{v}] = \text{manoeuvre}(\mu, R, t, \delta\alpha, \dots)$$

In this expression, the gravitational parameter μ , the orbital radius R , the maneuver time, and the differential orbital elements occur. This choice of input parameters is to some extent arbitrary. The maneuver calculations all rely on the orbital rate n rather than directly on the gravitational parameter and the orbital radius. This means that the calculation of the orbital rate could also be removed from the maneuver calculation and moved to some preprocessing function.

$$n = \sqrt{\frac{\mu}{R^3}}$$

In this case, the calculation required to obtain the orbital rate is fairly simple, and the computational cost of performing this operation within the maneuver calculation is small. On the other hand, the orbital rate is also used in trajectory propagation functions that use the output of the maneuver calculations to propagate a reference trajectory. Placing the calculation of the orbital rate inside a preprocessing function ensures that the orbital rate is used consistently throughout the whole guidance and ensures that the number of mathematical operations inside the guidance is reduced. For more complex types of intermediate results, it may be worth the effort to ensure that these calculations are performed only once.

The maneuver calculation algorithms are broadly similar in the sense that all of them output at the very least a ΔV . On the other hand, some of the impulsive maneuver calculation algorithms also output the time of application of the maneuver. It is recommended to ensure that all maneuver calculation functions have broadly similar interfaces.



Guidance implementation best practices

The following recommendations are provided for the implementation of guidance functions. These recommendations are based on the overall suggestions that have been provided throughout this chapter.

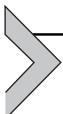
- Ensure consistency of dynamics and state representations with N and C.
- Define the guidance architecture.
- Define clear interfaces with the other functional elements in the GNC software. Ensure that the interfaces are frozen early in the design and ensure that any changes are propagated throughout the full GNC.
- Perform early integration of the full GNC to ensure that the interfaces are consistent and that the full GNC runs correctly without errors (even if the full GNC is not complete yet).
- Ensure that all code contains a commented description of the function behavior, its inputs, and its outputs.
- Ensure that the function is clearly commented to ensure that other developers can easily understand what each line of code does.
- Break down functions into smaller function blocks.
- Reuse library functions whenever possible.
- Perform unit testing on smaller functions to ensure these functions behave as expected. Example of unit testing would be to perform numerical differentiation to obtain the rate of change of certain parameters to check the analytical expressions. For example, (angular) velocity can be checked by numerically differentiating the corresponding position vector/quaternion.
- Ensure that the accuracy of the guidance dynamics models is sufficient to achieve the mission goals.
- Ensure that the accuracy of the guidance dynamics models do not exceed the accuracy that can be provided by the rest of the functions in the GNC system.

References

- [1] D.K. Geller, Orbital rendezvous: when is autonomy required? *Journal of Guidance, Control, and Dynamics* 30 (4) (2007) 974–981.
- [2] J.R. Wertz, W.J. Larson, D. Klungle, in: *Space Mission Analysis and Design*, vol 8, Torrance, Microcosm, 1999.
- [3] <https://www.h2020-ergo.eu/project/background-on-autonomy-software-frameworks/autonomy-in-space-systems/>.
- [4] P. Grandjean, T. Pesquet, A.M.M. Muxi, M.C. Charnneau, What on-board autonomy means for ground operations: an autonomy demonstrator conceptual design, in: *Space OPS 2004 Conference*, 2004, p. 267.

- [5] M. Chernick, S. D'Amico, New closed-form solutions for optimal impulsive control of spacecraft relative motion, *Journal of Guidance, Control, and Dynamics* 41 (2) (2018) 301–319.
- [6] J.L. Junkins, P. Singla, How nonlinear is it? A tutorial on nonlinearity of orbit and attitude dynamics, *Journal of the Astronautical Sciences* 52 (1) (2004) 7–60.
- [7] M.D. Shuster, A survey of attitude representations, *Navigation* 8 (9) (1993) 439–517.
- [8] J.L. Synge, A. Schild, *Tensor calculus*, in: *Tensor Calculus*, University of Toronto Press, 2020.
- [9] P.C. Hughes, *Spacecraft Attitude Dynamics*, Courier Corporation, 2012.
- [10] J. Sola, Quaternion Kinematics for the Error-State Kalman Filter, 2017 arXiv preprint arXiv:1711.02508.
- [11] J.B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*, Princeton University Press, 1999.
- [12] H. Schaub, J.L. Junkins, Stereographic orientation parameters for attitude dynamics: a generalization of the Rodrigues parameters, *Journal of the Astronautical Sciences* 44 (1) (1996) 1–19.
- [13] G.R. Hintz, Survey of orbit element sets, *Journal of Guidance, Control, and Dynamics* 31 (3) (2008) 785–790.
- [14] J. Sullivan, S. Grimberg, S. D'Amico, Comprehensive survey and assessment of spacecraft relative motion dynamics models, *Journal of Guidance, Control, and Dynamics* 40 (8) (2017) 1837–1859.
- [15] W. Fleming, R. Rishel, The simplest problem in calculus of variations, in: *Deterministic and Stochastic Optimal Control*, Springer, New York, NY, 1975, pp. 1–19.
- [16] W.H. Fleming, R.W. Rishel, *Deterministic and Stochastic Optimal Control*, vol 1, Springer Science & Business Media, 2012.
- [17] A.E. Bryson, Y.C. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*, Routledge, 2018.
- [18] D. Liberzon, *Switching in Systems and Control*, vol 190, Birkhauser, Boston, 2003.
- [19] B.A. Conway (Ed.), *Spacecraft Trajectory Optimization*, vol 29, Cambridge University Press, 2010.
- [20] J.T. Betts, Survey of numerical methods for trajectory optimization, *Journal of Guidance, Control, and Dynamics* 21 (2) (1998) 193–207.
- [21] D.F. Lawden, Interplanetary rocket trajectories, *Advances in Space Science* 1 (1959) 1–53.
- [22] D.J. Jezewski, H.L. Rozendaal, An efficient method for calculating optimal free-space n-impulse trajectories, *AIAA Journal* 6 (11) (1968) 2160–2165.
- [23] B. Fornberg, *A Practical Guide to Pseudospectral Methods* (No. 1), Cambridge University Press, 1998.
- [24] K.C. Howell, H.J. Pernicka, Numerical determination of Lissajous trajectories in the restricted three-body problem, *Celestial Mechanics* 41 (1) (1987) 107–124.
- [25] O. Montenbruck, E. Gill, F. Lutze, Satellite orbits: models, methods, and applications, *Applied Mechanics Reviews* 55 (2) (2002) B27–B28.
- [26] K.T. Alfriend, S.R. Vadali, P. Gurfil, J.P. How, L. Breger, *Spacecraft Formation Flying: Dynamics, Control and Navigation*, vol 2, Elsevier, 2009.
- [27] K. Yamanaka, F. Ankersen, New state transition matrix for relative motion on an arbitrary elliptical orbit, *Journal of Guidance, Control, and Dynamics* 25 (1) (2002) 60–66.
- [28] K. Yamada, M. Kimura, T. Shima, S. Yoshikawa, New state transition matrix for formation flying in J2-perturbed elliptic orbits, *Journal of Guidance, Control, and Dynamics* 35 (2) (2012) 536–547.
- [29] A.W. Koenig, T. Guffanti, S. D'Amico, New state transition matrices for relative motion of spacecraft formations in perturbed orbits, in: *AIAA/AAS Astrodynamics Specialist Conference*, 2016, p. 5635.

- [30] D.W. Gim, K.T. Alfriend, State transition matrix of relative motion for the perturbed noncircular reference orbit, *Journal of Guidance, Control, and Dynamics* 26 (6) (2003) 956–971.
- [31] C.D. Murray, S.F. Dermott, *Solar System Dynamics*, Cambridge University Press, 1999.
- [32] T.V. Peters, R. Noomen, Linear cotangential transfers and safe orbits for elliptic orbit rendezvous, *Journal of Guidance, Control, and Dynamics* 44 (4) (2020) 732–748.
- [33] T.V. Peters, R. Noomen, P. Colmenarejo, Analytical solutions to two-impulse non-drifting transfer problems for rendezvous in elliptical orbits, *Journal of Guidance, Control, and Dynamics* 37 (3) (2014) 775–788.
- [34] M.J. Kim, M.S. Kim, S.Y. Shin, A general construction scheme for unit quaternion curves with simple high order derivatives, in: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, September 1995, pp. 369–376.
- [35] M. Kelly, An introduction to trajectory optimization: how to do your own direct collocation, *SIAM Review* 59 (4) (2017) 849–904.



Navigation

**Vincenzo Pesce¹, Pablo Hermosin², Aureliano Rivolta³,
Shyam Bhaskaran⁴, Stefano Silvestrini⁵, Andrea Colagrossi⁵**

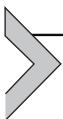
¹Airbus D&S Advanced Studies, Toulouse, France

²Deimos Space, Madrid, Spain

³D-Orbit, Fino Mornasco, Italy

⁴NASA Jet Propulsion Laboratory, Pasadena, CA, United States

⁵Politecnico di Milano, Milan, Italy



What is navigation?

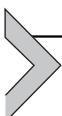
The next concept of the guidance, navigation, and control (GNC) chain to be introduced is navigation. Going back to our person with a plan on where to go and what to do, it is of primary importance for the success of the task to know and understand the current location. For human beings, it is natural and relatively effortless to establish where they are. In general, they can process sensorial information, like tactile and visual perception along with previous knowledge and experience, to correctly know their location. For example, our memory and our capacity of virtual abstraction allow us to establish how far from a wall we are with a certain degree of uncertainty. Similarly, our vestibular system provides our brain with information about our motion, allowing a person to discern the body orientation and spatial position. In addition, we can consider the localization task of a person in its environment, in various human ages. Before the establishment of cartography, people would find aid in the surroundings to locate themselves (e.g., moss or Stella Polaris to identify the North). The discovery of the compass and its use strongly improved human navigation ability since the 11th century. Most recently, with the advent of global positioning system (GPS) and modern smartphones, the localization task has become very easy and extremely precise. These examples clarify and simplify the concept of navigation, i.e., processing sensorial information together with experience and knowledge of the environment to perform a localization task. This definition is valid also for spacecraft navigation. In fact, sensors and models of the environment (i.e., dynamical models) are used to estimate the state, rotational and/or translational, of the spacecraft. This process can either be done on-board, commonly called autonomous navigation, or on-ground

by a team of trained specialists. The availability of sensors, their accuracy, and modeling are fundamental for a precise state estimation also for a spacecraft. Another important role is played by the knowledge of the environment and the capacity of describing the spacecraft motion with equations. In fact, the accuracy of a dynamical model actively influences the navigation performance. It is important to underline that both sensors and dynamical models are giving uncertain and stochastic inputs. The goal of the navigation is to process and combine in a clever way this information in order to find a precise and accurate, but still uncertain, spacecraft's state estimation. Navigation has to provide the spacecraft's state information to the control functions for a comparison with the guidance output. Clearly, if the state is not correctly estimated by the navigation algorithms, this error immediately leads to a wrong control action on the spacecraft and, therefore, to a large discrepancy with respect to the initial plan. With these notions in mind, we have the basics to start exploring the technicalities of navigation algorithms and functions.

This chapter is composed by the following sections:

- *Sequential and parameters estimation filtering techniques.* In this section, the working principle and derivation of the main sequential filters for space application are detailed. Linear and nonlinear estimation techniques are introduced for standard state estimation. Moreover, these formulations are extended to the special case of parameters estimation.
- *Batch estimation filtering techniques.* This section starts with an explanation of the least squares filter, followed by the inclusion of dynamic effects and observation errors and, finally, the concepts of data fusion with a priori information, which constitute the base for the different filters that will be explained afterward. A detailed section on the typical problems that can arise during trajectory reconstruction is also included.
- *Absolute orbit navigation.* This section has the purpose of introducing the basics of global navigation satellite system (GNSS) and its observables and how to use them to perform absolute spacecraft navigation. Furthermore, it introduces absolute navigation exploiting pulsars line-of-sight (LOS) measurements, particularly useful for interplanetary scenarios. Finally, the most important techniques and instruments to perform ground-based orbit determination (OD) are introduced.
- *Absolute attitude navigation.* Common attitude estimation techniques are presented and detailed in this section.

- *Relative navigation.* This section discusses the basic principle of relative navigation offering an overview of the possible approaches that can be adopted while designing a navigation filter for this kind of application.
- *Image processing techniques.* The most important algorithms to perform image processing useful to spacecraft navigation are detailed in this section. Segmentation techniques, 2D, 3D, and multiview methods are introduced with a description on how and when to use this kind of measurements in a navigation filter.
- *Navigation budgets.* In this section, the influence of the navigation on the overall GNC chain is described and the estimation accuracy can affect the other GNC components.
- *Navigation implementation best practice.* A series of best practice tips and checks is collected in this section, giving a more practical perspective on the implementation of navigation algorithms.



On-board versus ground-based navigation

Spacecraft navigation involves several different elements (e.g., sensors, actuators, dynamical model, measurement model, filter), but the navigation filter is undoubtedly its core. In an ideal world, the state of a certain quantity could be known exactly by relying on perfect dynamical model or extraordinarily precise sensors. In real world, the same state is described by approximate models and the available sensors produce measurements with not negligible errors. Our only option is to combine, in the best possible way, these uncertain quantities. In statistic and dynamical system theory, a filter combines a set of uncertain measurements and a prediction of the dynamical evolution of the spacecraft (according to an uncertain dynamical model) to obtain the best possible estimate of the spacecraft's state vector. Two main approaches are usually adopted when dealing with spacecraft navigation:

- *Sequential filtering.* The state vector is sequentially updated at each time step given a single-step measurement and the state predicted by the dynamical model. Sequential filters are usually adopted for on-board spacecraft navigation.
- *Batch filtering.* The state vector is updated taking into account a set of measurements collected during a given time window. This approach is commonly used for on-ground spacecraft navigation.

The difference between sequential and batch filtering is not the only distinctive factor between on-board and on-ground spacecraft navigation.

In general, the following aspects can be considered as differentiating characteristics between the two approaches:

- *Precision.* On-ground navigation techniques are usually more accurate since more precise dynamical and measurement models are available. In fact, the computational power is not constrained by stringent hardware limitations typical of space missions. As defined above, typically on-ground solutions are associated to batch filters that are intended to reconstruct a complete spacecraft state during a trajectory arc. The reconstruction of the trajectory usually involves a huge amount of data coming from different sensors that are merged to obtain the navigation solution. The fact that large amount of data can be employed leads to higher accuracy in the solution. In contrast, on-board navigation is constrained by the data that can be processed in real time by the on-board system. Therefore, the solution associated with on-board filtering techniques is usually less accurate than the reconstructed trajectory on-ground.
- *Communication delays.* The delay in the communication with a spacecraft is also a key factor when deciding if the navigation solution can be obtained on-board or if it is better to rely on on-ground estimations to update the spacecraft state. For instance, during interplanetary legs of a mission to an outer planet, relying on on-ground OD solutions is not a problem since the navigation solution is needed at a very low frequency and large delays in communication are not critical. On the other hand, in a mission involving close proximity operations around an asteroid such as landing, low altitude flyby, or sample recollection, it may be more interesting to opt for a navigation solution calculated on-board, since the delay in communications would be unacceptable for such a critical phase. In these situations, having the navigation calculated on-ground would not allow for the fast response required from the environment: the reaction time when performing an avoidance maneuver during a landing phase might be in the order of seconds whereas the delay in communications with Earth would be, best-case scenario, in the order of minutes in deep space missions.
- *Cost.* Ground operations and, therefore, on-ground navigation have a cost which is nonnegligible in a space mission. On the other hand, on-board navigation usually implies costs related to the technological development of a dedicated navigation solution in terms of additional hardware and algorithms.

- *Spacecraft hardware complexity.* On-board navigation usually relies on additional sensors that have to guarantee the sufficient accuracy to perform navigation. As an example, we can mention cameras for approaching a small body [1]. The presence of additional sensors increases the hardware complexity of the spacecraft and directly influences the spacecraft design at system level.

When deciding between on-board or on-ground OD, there is not an absolute answer and a trade-off, taking into account all the presented relevant factors, needs to be carried out.



Sequential filters

Sequential filters belong to a category of estimators that provide a state vector estimate, using measurements available at a specific instant of time. Sequential filters are typically used for on-board spacecraft navigation, especially when dealing with relative scenarios, e.g., formation flying (FF), rendezvous, and landing. Due to the limited computational power available on-board, a single set of measurements, corresponding to the last available time step, is processed instead of a “batch” of consecutive measurements.

Given an initial state vector, the knowledge of its dynamical evolution is used to predict the state vector at a next step. This predicted state is then updated given the measurements collected at the current time step. The measurements predicted through the measurement model (also called observation model) are compared to the actual measurements to produce the filter estimate. The best estimate is finally fed as input for the propagation at the following step, and the process can sequentially continue. Fig. 9.1 shows the high-level schematic of a generic sequential estimator.

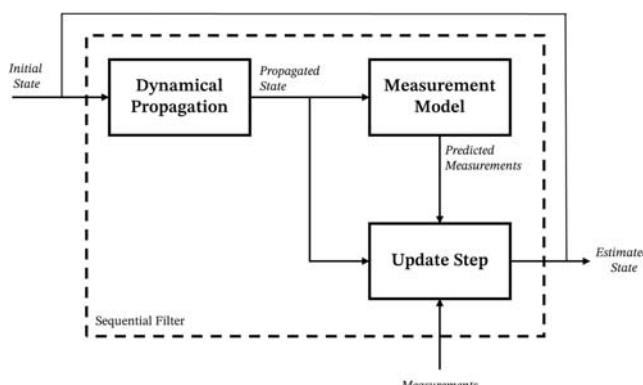


Figure 9.1 Sequential filter—high-level schematic.

This chapter starts with a conceptual explanation of the working principle of sequential filters, followed by a theoretical discussion on the most common sequential estimators.

Working principle

Before presenting the mathematical formulation of the most common sequential filters, a practical discussion on the filtering steps is deemed necessary. The working principle of sequential filters can be explained by considering a simple example. Let's take a spacecraft orbiting a distant exoplanet. The spacecraft has to determine its 2D position $\mathbf{x} = [X, Y]$ (for sake of simplicity), using noisy measurements and relying on an uncertain description of its orbital motion. It is important to clarify that the information on the exact position of the spacecraft is not available; therefore, its state can be described only using a probability distribution function (e.g., a Gaussian distribution) with mean μ , being its best estimate, and variance σ , representing its uncertainty, as shown in Fig. 9.2.

The filtering process is composed by two main steps:

- *Prediction step.* Using the best knowledge of the exoplanet environment, it is possible to propagate the best state estimate (distribution mean) and the state uncertainty (distribution covariance – please note that while variance is a statistical quantity referring to a single variable, covariance provides information on how two variables are related to one another) using a dynamical model $f(\mathbf{x})$. It is important, while propagating the state uncertainty, to consider how the unmodeled effects can affect this

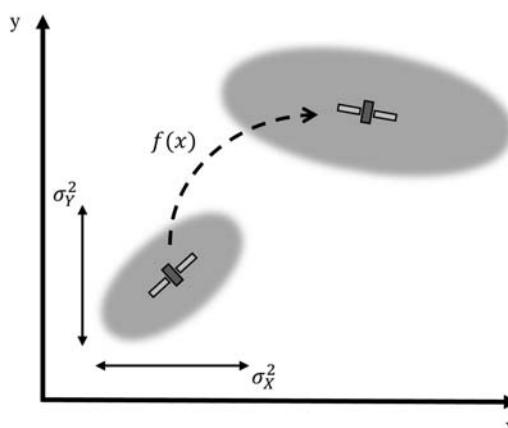


Figure 9.2 Sequential filter—prediction step.

propagation. This is done by adding to the propagated covariance the so called “process noise” \mathbf{Q} . This step is graphically depicted in Fig. 9.2. After this step, a new predicted state is available with its associated covariance that is usually larger than the initial one due to the unmodeled dynamical effects.

- *Update step.* Once the predicted state is available, it is possible to compute the expected measurement value. In this simple example, the predicted measurement is directly the predicted position, but, in general, the expected measurements have to be obtained by relying on a measurement model, $h(\mathbf{x})$. The measurement model (or sensor model) is a function that provides the expected sensor output given a certain state. The expected (or predicted) measurements and the actual sensor readings, \mathbf{y} , along with their uncertainties, are then merged, given the propagated state, to find the best estimate of the state. The way two probability distributions are combined is by multiplying them together, obtaining the overlapping region of the two probability distributions as in Fig. 9.3.

The result of this final step is another probability distribution that is a lot more precise than either of the two previous estimates. The mean of this distribution is the most likely estimate and, therefore, the best guess of the spacecraft state.

Sequential filters for spacecraft navigation

The previous description of the working principle of a sequential estimator is general, but several specific algorithmic solutions exist to perform the *prediction* and *update* steps of a sequential estimator. In this section, the most common sequential estimators used for spacecraft navigation are described.

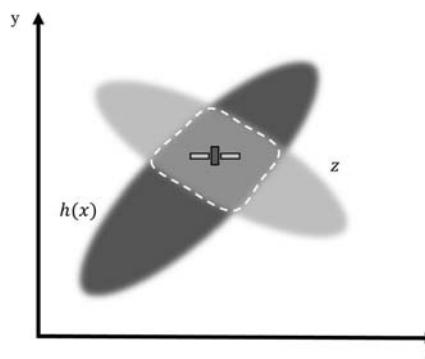


Figure 9.3 Sequential filter—update step.

Kalman filter

The most widely used and known filtering technique is certainly the Kalman filter (KF) [2,3]. It takes its name from Rudolf E. Kálmán, a pioneer of modern estimation theory, and it has been extensively used in space applications, including the navigation filter of the Apollo project. The main assumption behind the KF is that both state evolution and measurements can be described using linear models. This is usually not the case for real-life applications, but a linear approximation may be sufficient to reach the required estimation accuracy. In this paragraph, the standard form of discrete-time KF is presented.

Let's consider a linear discrete-time system:

$$\mathbf{x}_k = \mathbf{F}_{k-1} \mathbf{x}_{k-1} + \mathbf{G}_{k-1} \mathbf{u}_{k-1} + \mathbf{w}_{k-1}$$

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

In the above equation, \mathbf{x}_k is state vector, representing the variables that have to be estimated, \mathbf{u}_k is the control input applied to the system, and \mathbf{y}_k is the sensors' output. The quantities \mathbf{w}_k and \mathbf{v}_k represent the process and measurement noises, with associated covariance matrices \mathbf{Q}_k and \mathbf{R}_k . They describe the uncertainty related to the adopted dynamical and measurement models. Finally, \mathbf{F}_k , \mathbf{G}_k , and \mathbf{H}_k are the state-transition, the model-input, and the measurement model matrices. Given the estimate of the state at the previous step $\hat{\mathbf{x}}_{k-1}$, the best estimate at the current step is obtained by first propagating the state using the dynamical model:

$$\hat{\mathbf{x}}_k^- = \mathbf{F}_{k-1} \hat{\mathbf{x}}_{k-1}^+ + \mathbf{G}_{k-1} \mathbf{u}_{k-1}.$$

This is the *state prediction* step of the KF. Please note that $\hat{\mathbf{x}}^+$ represents the a-posteriori best estimate: the expected value of \mathbf{x}_t , conditioned on all the measurements up to time t . Instead, $\hat{\mathbf{x}}^-$ is the a priori estimate: the expected value of \mathbf{x}_t conditioned on all the measurements up to time $t-1$. Using classical probability theory [4], the covariance of a linear discrete-time system can be propagated as:

$$\mathbf{P}_k^- = \mathbf{F}_{k-1} \mathbf{P}_{k-1}^+ \mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}.$$

This is the *covariance prediction* step of the KF. Recalling the concepts introduced before, the prediction step provides a probability distribution of a predicted state, taking into account the dynamical model uncertainties and all the unmodeled effects represented by the matrix \mathbf{Q}_{k-1} .

The next step is to derive the *update* step, according to the available measurements \mathbf{y}_k . Without entering into the theoretical details of the derivation, the *state and covariance update* step can be expressed as:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-)$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k-1}^-$$

with \mathbf{K}_k being the KF gain matrix, derived as:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}.$$

It is worth underlying that there is not any assumption on the Gaussian nature of process and measurements noises, \mathbf{w}_k and \mathbf{v}_k . It results that the KF is an optimal filter when the noise is Gaussian, zero-mean, uncorrelated, and white [3], but it is still the best linear estimator if the Gaussian assumption does not hold. To strengthen once more the working principle of KFs, a graphical example of prediction and update steps of a one-dimensional state distribution is represented in Fig. 9.4.

In the figure on the left side, the prediction step is depicted. $\hat{\mathbf{x}}_{k-1}^+$, the estimate of the state at time $k-1$ and its corresponding covariance \mathbf{P}_{k-1}^+ are propagated according to the dynamical model \mathbf{F}_{k-1} and its uncertainty \mathbf{Q}_{k-1} . This step leads to a new predicted state $\hat{\mathbf{x}}_k^-$ with its associated covariance \mathbf{P}_k^- . The effects of the dynamical uncertainties \mathbf{Q}_{k-1} are highlighted by drawing the covariance with and without this term. As expected, the uncertainties and all the unmodeled effects of the dynamical model spread the curve and, therefore, increase the overall state uncertainty. In the right figure, instead, the update step is represented. The predicted state $\hat{\mathbf{x}}_k^-$ and its covariance \mathbf{P}_k^- are combined with the measurements \mathbf{y}_k and the associated

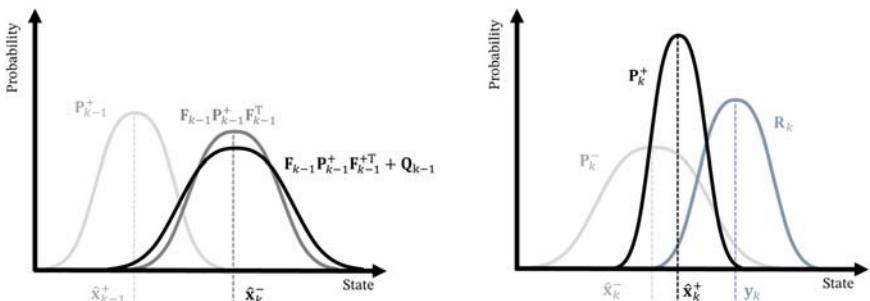


Figure 9.4 Kalman filter—1D example of prediction (left) and update (right) steps.

uncertainty \mathbf{R}_k to obtain the best state estimate $\hat{\mathbf{x}}_k^+$ and state covariance \mathbf{P}_k^+ . The state estimate covariance \mathbf{P}_k^+ is the result of the combination of two gaussian curves, namely the predicted covariance \mathbf{P}_k^- and the measurement covariance \mathbf{R}_k .

$H\infty$ filter

For linear systems with process and measurement noise represented by a zero-mean Gaussian distribution, KF is the minimum variance estimator. In other words, the KF represents the optimal estimator when the model is linear and the noise Gaussian. However, to satisfy these assumptions and to guarantee a good tuning of the filter, the mean of \mathbf{w}_k and \mathbf{v}_k and their covariance \mathbf{Q}_k and \mathbf{R}_k have to be known. Properly tuning these parameters is usually a difficult task. This is especially true for real applications where the model uncertainty is unknown and difficult to estimate. In the case of non-Gaussian noise, nonlinear system or if the tuning is off nominal, a filter that minimizes the worst-case estimation error rather than the variance of the estimation error could outperform the KF. This kind of filter is called $H\infty$ filter or also minimax filter. It minimizes the 1-norm of the estimation error, and it does not make any assumption about the statistics of the process and measurement noise [5]. The formulation of the $H\infty$ filter is very similar to the one of the KF. In fact, the prediction step is performed in the same way, but, instead, a slightly different expression of the Kalman gain \mathbf{K}_k is derived. In particular, \mathbf{K}_k has to be chosen such that $\|\mathbf{T}_{ew}\|_\infty < \frac{1}{\vartheta}$, where \mathbf{T}_{ew} represents the difference between the predicted and real state and ϑ is a tuning parameter. The derived expression for \mathbf{K}_k is:

$$\mathbf{K}_k = \mathbf{P}_k^- [\mathbf{I} - \vartheta \mathbf{P}_k^- + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \mathbf{P}_k^-]^{-1} \mathbf{H}_k^T \mathbf{R}_k^{-1}.$$

Please note that the performance of the $H\infty$ filter is more robust to model and sensor uncertainties but usually worse with respect to KF if working in nominal condition. Moreover, it is sensitive to the selection of the tuning parameter ϑ . This kind of filter is the best option when dealing with very uncertain systems and time-varying dynamics, offering a robust alternative to KF.

Extended Kalman filter

KF and $H\infty$ filter are linear filters that can be used when the dynamics underlying the investigated phenomenon is linear. However, linear systems are only approximations of more complex nonlinear dynamics. When the

behavior of a system cannot be described by a linear function, linear filters are no longer adequate. In all these cases, nonlinear estimators, such as the extended Kalman filter (EKF), shall be employed. A common approach to nonlinear state estimation is to use a modified version of the standard KF to cope with the nonlinearities in the dynamical equations. This is the so-called EKF [6]. The idea behind the EKF is straightforward. In practice, the nonlinear system is linearized around the current state estimate, and the following state estimate is obtained from the linearized system. In this section, the discrete-time EKF is presented. Let's consider the nonlinear model:

$$\mathbf{x}_k = f_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$

$$\mathbf{y}_k = h_k(\mathbf{x}_k, \mathbf{v}_k)$$

with, as before, \mathbf{x}_k the state vector, \mathbf{u}_k the control input, \mathbf{w}_k and the \mathbf{v}_k the process and measurement noises, with associated covariance matrices, \mathbf{Q}_k and \mathbf{R}_k , and \mathbf{y}_k the measurement output. f_k and h_k are the functions describing the state dynamics and observation model. Performing local linearization around the state estimate, the following algorithm is obtained:

$$\hat{\mathbf{x}}_k^- = f_{k-1}(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1}, 0)$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1} \mathbf{P}_{k-1}^+ \mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - h(\hat{\mathbf{x}}_k^-, 0))$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k-1}^-$$

where the state-transition matrix and observation covariance are derived as:

$$\mathbf{F}_{k-1} = \frac{\partial f_{k-1}}{\partial \mathbf{x}} \Big|_{\hat{\mathbf{x}}_{k-1}^+}$$

$$\mathbf{H}_{k-1} = \frac{\partial h_{k-1}}{\partial \mathbf{x}} \Big|_{\hat{\mathbf{x}}_k^-}.$$

Please note that the propagation of the state is still nonlinear and the linearized state-transition matrices \mathbf{F}_{k-1} and \mathbf{H}_{k-1} are exploited for covariance propagation. The EKF is the standard approach for state estimation and navigation filters. However, the main drawback of this kind of filter is that, relying on linearization for state covariance and mean propagation, it is

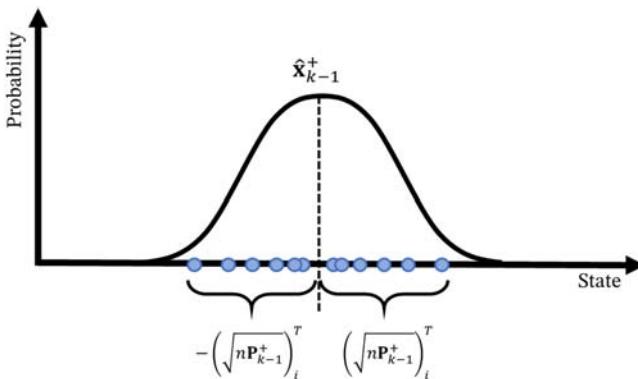


Figure 9.5 Sigma points distribution.

usually difficult to tune. Nonlinearities may introduce bias in dynamical or measurement equations by altering the noise distributions, yielding to sub-optimal performance or divergence. This is especially true for highly nonlinear systems.

Unscented Kalman filter

The main alternative to the EKF is the unscented Kalman filter (UKF) [7]. This filter avoids the local linearization used by the EKF, and it uses an unscented transformation to propagate the mean and covariance of the state. Unscented transform better describes the result of a given nonlinear transformation applied to a probability distribution by performing a weighted sampling to compute mean and covariance. In fact, an unscented transform does not use the mean and covariance of a Gaussian distribution to perform a nonlinear transformation, but it exploits a set of points extracted from a distribution with the same mean and covariance. In this way, the full nonlinear function can be applied directly to this set of points, also called sigma points. Therefore, each sigma point is propagated in a nonlinear fashion and a new set of propagated points is obtained. This new set represents a new distribution, and, therefore, a new mean and covariance can be computed without relying on local linearization and on linearized state-transition matrices. A very simple 2D example is depicted in Fig. 9.6. Let's consider a nonlinear system with additive noise:

$$\mathbf{x}_k = f_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}$$

$$\mathbf{y}_k = h_k(\mathbf{x}_k) + \mathbf{v}_k$$

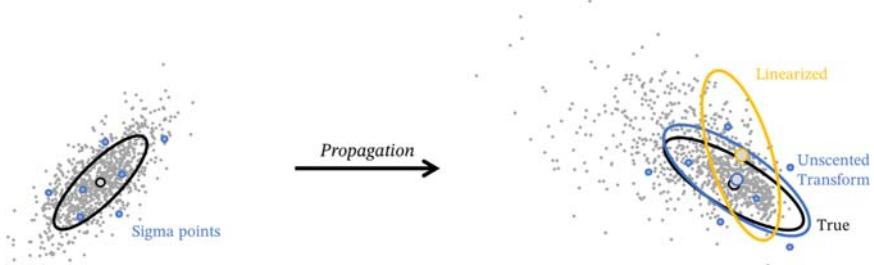


Figure 9.6 Unscented Transform—2D example of state and covariance propagation and approximation.

The prediction step is performed by considering $2n$ sigma points, where n is the dimension of the state vector. The sigma points $\tilde{\mathbf{x}}^{(i)}$ are selected according to:

$$\hat{\mathbf{x}}_{k-1}^{(i)} = \hat{\mathbf{x}}_{k-1}^+ + \tilde{\mathbf{x}}^{(i)} \quad i = 1, \dots, 2n$$

$$\tilde{\mathbf{x}}^{(i)} = \left(\sqrt{n \mathbf{P}_{k-1}^+} \right)_i^T \quad i = 1, \dots, n$$

$$\tilde{\mathbf{x}}^{(n+i)} = - \left(\sqrt{n \mathbf{P}_{k-1}^+} \right)_i^T \quad i = 1, \dots, n$$

Please note that the sigma points are extracted considering two subsets as in Fig. 9.5.

The set of sigma points is then propagated using the dynamical model:

$$\hat{\mathbf{x}}_k^{(i)} = f_{k-1}(\hat{\mathbf{x}}_{k-1}^{(i)}, \mathbf{u}_{k-1}).$$

Then, these $2n$ vectors can be combined to obtain the a priori state estimate and its covariance:

$$\hat{\mathbf{x}}_k^- = \frac{1}{2n} \sum_{i=1}^{2n} \hat{\mathbf{x}}_k^{(i)}$$

$$\mathbf{P}_k^- = \frac{1}{2n} \sum_{i=1}^{2n} (\hat{\mathbf{x}}_k^{(i)} - \hat{\mathbf{x}}_k^-) (\hat{\mathbf{x}}_k^{(i)} - \hat{\mathbf{x}}_k^-)^T + \mathbf{Q}_{k-1}.$$

In a similar way, the update step is performed. First, other $2n$ sigma points are selected based on the a priori estimate:

$$\hat{\mathbf{x}}_{k-1}^{(i)} = \hat{\mathbf{x}}_k^- + \tilde{\mathbf{x}}^{(i)} \quad i = 1, \dots, 2n$$

$$\tilde{\mathbf{x}}^{(i)} = (\sqrt{n\mathbf{P}_k^-})^T \quad i = 1, \dots, n$$

$$\tilde{\mathbf{x}}^{(n+i)} = -(\sqrt{n\mathbf{P}_k^-})^T_i \quad i = 1, \dots, n$$

Note that for a faster algorithm, the sigma points obtained before can be used for this step, but this will result in degraded performance. At this point, the observation model is used to predict the measurements:

$$\hat{\mathbf{y}}_k^{(i)} = h_k(\hat{\mathbf{x}}_k^{(i)}).$$

Finally, the predicted measurement covariance and the cross covariance can be computed:

$$\begin{aligned} \mathbf{P}_y &= \frac{1}{2n} \sum_{i=1}^{2n} (\hat{\mathbf{y}}_k^{(i)} - \hat{\mathbf{y}}_k^-) (\hat{\mathbf{y}}_k^{(i)} - \hat{\mathbf{y}}_k^-)^T + \mathbf{R}_{k-1} \\ \mathbf{P}_{xy} &= \frac{1}{2n} \sum_{i=1}^{2n} (\hat{\mathbf{x}}_k^{(i)} - \hat{\mathbf{x}}_k^-) (\hat{\mathbf{y}}_k^{(i)} - \hat{\mathbf{y}}_k^-)^T. \end{aligned}$$

This leads to the a posteriori state estimate:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k (\mathbf{y}_k - \hat{\mathbf{y}}_k)$$

$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_y \mathbf{K}_k^T$$

$$\mathbf{K}_k = \mathbf{P}_{xy} \mathbf{P}_y^{-1}.$$

The presented formulation is valid under the assumption of additive noise. If this assumption is not valid, a different formulation has to be derived [5]. The UKF presents some advantages with respect to the EKF and overcome its intrinsic limitations. In fact, the computation of Jacobians is not required in this case. However, computing the propagation of $2n$ sigma points can be computationally demanding in case of complex system dynamics. Usually, UKF is preferred when the adopted dynamical model is not very complex.

Particle filter

Particle filters (PFs) represent an alternative to EKF and UKF. They were first introduced in 1993 [8] with the name of bootstrap filter. The key idea underlying the PF is to approximate the filtering density function as a weighted set of samples, also called particles. Its representation is fundamentally different from the one used in the KF, where a specific functional form of the density function is assumed, and the estimate is then represented by the quantities (the mean and the covariance) parameterizing this density. In the PF, the filtering density is represented as a set of random samples approximately distributed according to this density. PF's working principle is similar to the one of the UKF, since it is based on the propagation of a set of points via known nonlinear equations and, successively, on the combination of the results to estimate the state and state uncertainty. However, in the PF, the points are chosen according to a stochastic sampling, whereas in the UKF, the points are chosen deterministically. In fact, UKF relies on the unscented transform and on a deterministic choice of the sigma points. Moreover, in PF, the posterior distribution of the state is not assumed to be Gaussian as done in a classical UKF where mean and covariance are employed to describe the posterior Gaussian distribution. For these reasons, PF usually relies on a much higher number of points if compared to the UKF, resulting in a higher computational load. As before, a generic nonlinear system can be described as:

$$\mathbf{x}_k = f_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})$$

$$\mathbf{y}_k = h_k(\mathbf{x}_k, \mathbf{v}_k)$$

For PF, the Bayesian approach to nonlinear state estimation is introduced. The aim of this strategy is to compute or approximate the posterior distribution of the state, given the observations. In particular, the Bayesian recursive solution to compute the posterior distribution ($p(\mathbf{x}_k | \mathbf{y}_{1:k})$) of the state, given past observation, is expressed as:

$$\text{Initialization: } p(\mathbf{x}_0 | \mathbf{Y}_{k-1}) = p(\mathbf{x}_0)$$

$$\text{Prediction: } p(\mathbf{x}_k | \mathbf{Y}_{k-1}) = \int p(\mathbf{x}_k | \mathbf{x}_{k-1}) p(\mathbf{x}_{k-1} | \mathbf{Y}_{k-1}) d\mathbf{x}_{k-1}$$

$$\text{Correction: } p(\mathbf{x}_k | \mathbf{Y}_k) = \frac{p(\mathbf{y}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{Y}_{k-1})}{\int p(\mathbf{y}_k | \mathbf{x}_{k-1}) p(\mathbf{x}_k | \mathbf{Y}_{k-1})}$$

with $\mathbf{Y}_k = \mathbf{y}_{1:k}$.

For a general case, there isn't an explicit solution for this integral. However, for a linear system with Gaussian noises, the classical KF recursive form provides the solution for the presented *Bayesian* problem. For a generic nonlinear system, with non-Gaussian noises, it is necessary to rely on numerical approximations. PFs offer a tool to approximately solve the recursion for a generic system. In particular, M random points (particles) are generated at the beginning of the estimation, based on the initial probability density function (pdf) of the state. In fact, it is reasonable to approximate the pdf as sum of δ and in particular:

$$p(\mathbf{x}_k | \mathbf{Y}_k) \simeq \hat{p}(\mathbf{x}_k | \mathbf{Y}_k) = \frac{1}{M} \sum_{i=1}^M \delta(\mathbf{x}_k - \bar{\mathbf{x}}_{k,i})$$

where $\bar{\mathbf{x}}_{k,i}$ are the particles extracted from the true conditional density. Therefore, the recursive algorithm can be derived as follows:

$$\text{Initialization: } p(\mathbf{x}_0) \simeq \frac{1}{M} \sum_{i=1}^M \delta(\mathbf{x}_0 - \bar{\mathbf{x}}_{0,i})$$

$$\text{Estimation : } \hat{p}(\mathbf{x}_{k-1} | \mathbf{Y}_{k-1}) \simeq \frac{1}{M} \sum_{i=1}^M \delta(\mathbf{x}_{k-1} - \bar{\mathbf{x}}_{k-1,i})$$

$$\text{Prediction: } p(\mathbf{x}_k | \mathbf{Y}_{k-1}) \simeq \frac{1}{M} \sum_{i=1}^M \delta(\mathbf{x}_k - \bar{\mathbf{x}}_{k,i})$$

$$\text{Correction: } p(\mathbf{x}_k | \mathbf{Y}_k) = \sum_{i=1}^M \mathbf{q}_i \delta(\mathbf{x}_k - \bar{\mathbf{x}}_{k,i})$$

$$\text{where } \bar{\mathbf{x}}_{k,i} = f(\bar{\mathbf{x}}_{k-1,i}, \bar{\mathbf{w}}_{k-1,i}), \bar{\mathbf{w}}_{k-1,i} \sim p(\mathbf{w}_{k-1}) \text{ and } \mathbf{q}_i = \frac{p(\mathbf{y}_k | \bar{\mathbf{x}}_{k,i})}{\sum_{i=1}^M p(\mathbf{y}_k | \bar{\mathbf{x}}_{k,i})}.$$

Note that the second set of points, $\bar{\mathbf{x}}_{k,i}$ is extracted from the already defined grid and the particles are propagated according to the nonlinear dynamics of the system f . The term \mathbf{q}_i indicates the relative probability of each particle, and it can be seen as a weight. The sum of all the weights is equal to 1. It is important to underline that, if resampling is not performed, the PF would end to a set of independent simulations, each one with its own weight or probability. Most likely, the so-called sample depletion would occur. This means that, not having any feedback from the observation,

the result would be to have all the weights tending to zero, except for one almost equal to 1. The high value of the weight does not mean that the estimated state is close to the real one but just that one sequence in the set of particles is more likely than the others. Resampling introduces the feedback from the observation and guarantees that the good state estimation does not disappear. A scheme graphically summarizing the main steps of a PF is displayed in Fig. 9.7.

Like the UKF, the PF method does not rely on any local linearization technique and do not have any constraint on the noise distribution. This flexibility can be very useful in several applications in which the EKF does not perform well (highly nonlinearities involved). However, all these advantages have a cost. In fact, the bottleneck of the PF is its computational load [9]. If we think at the EKF, only one function evaluation of $f(\mathbf{x}_k, \mathbf{w}_k)$ and $h(\mathbf{x}_k, \mathbf{v}_k)$ is required at each time step (note that if the Jacobian is not analytically available, more than one evaluation of these functions is needed). Per contra, M evaluations are needed with PF. This can become very demanding in systems with highly nonlinear and complicated dynamics

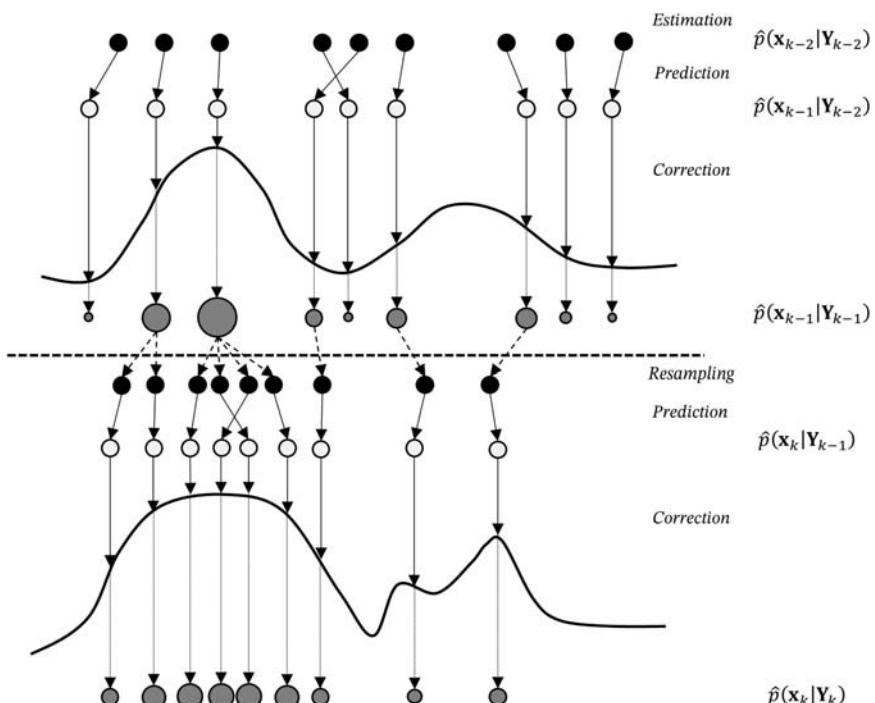


Figure 9.7 Particle filter—scheme.

and sampling with a high number of particles. This is one of the reasons why PF has never been practically implemented on any spacecraft. Summarizing, the PF overcomes the intrinsic limitations of the EKF and UKF, but the computational cost is much higher. In general, PF can be used when the Gaussian approximation of the posterior distribution is not acceptable (e.g., multimodal distributions) and when high computational power is available.

Parameters estimation

Sequential filters are typically used to estimate the states of a dynamical system. However, this kind of filters can be used also to estimate unknown parameters. This approach can be exploited for spacecraft navigation techniques when uncertain spacecraft parameters need to be estimated.

State augmentation for parameter estimation

The approach to estimate unknown parameters is similar to the one that it has already been presented for classical state estimation. However, the parameters to be estimated shall be included in the so-called augmented states. Let's consider the usual nonlinear model:

$$\begin{aligned}\mathbf{x}_k &= f_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \\ \mathbf{y}_k &= h_k(\mathbf{x}_k, \mathbf{v}_k).\end{aligned}$$

A parameter vector \mathbf{p} can be introduced in both state propagation and measurement prediction step as follows:

$$\begin{aligned}\mathbf{x}_k &= f_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}, \mathbf{p}) \\ \mathbf{y}_k &= h_k(\mathbf{x}_k, \mathbf{v}_k, \mathbf{p}).\end{aligned}$$

In order to be able to estimate the parameter vector \mathbf{p} , the state vector has to be augmented:

$$\mathbf{x}' = \begin{bmatrix} \mathbf{x}_k \\ \mathbf{p}_k \end{bmatrix}.$$

The augmented system model is now evolving according to the following equations:

$$\begin{aligned}\mathbf{x}'_k &= f_{k-1}(\mathbf{x}'_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \\ \mathbf{y}_k &= h_k(\mathbf{x}'_k, \mathbf{v}_k).\end{aligned}$$

Please note that if the parameter is constant, its first derivative will be set to 0 and this equation will describe its dynamical evolution according to f_{k-1} . All the presented filtering methods can be applied to solve this estimation problem, despite the augmentation of the state with the vector parameter.

Bias estimator

A classical parameter estimation case is represented by bias estimation problem. Spacecraft navigation usually involves the use of biased measurements. Bias is systematic error affecting a given measurement and in its simpler forms can be a random constant or a random ramp. When dealing with a bias modeled as a random constant, the estimation procedure is exactly the one described before: add the bias as state vector, include it in the measurement equation as an additional term to the considered measurement, and propagate a constant parameter (i.e., first derivative to 0 - refer to [Chapter 14](#) - Applicative GNC Cases and Examples for further details). Please keep in mind that, in order to converge, the initial state plus the associated initial covariance should be of the same order of magnitude as the expected bias (e.g., if a bias in the position measurement in the order of 1m is expected and the initial parameter value is 0, the δ of \mathbf{P}_0 should be at least 1m). On the other hand, if the bias is a random ramp (this may be the case of clock bias drift for GNSS signals), the dynamics has to be modified. In particular, the rate of change of the parameters is, in this case, a random constant, and, therefore, the second derivative of the parameter has to be set to 0. An example of bias estimation is shown in [Chapter 14](#) - Applicative GNC Cases and Examples.

Use of consider states—Schmidt—Kalman filter

In some cases, the inclusion of unknown parameters in the state vector may not be a recommended practice. In fact, even if a sequential estimator ignores the uncertainty that such parameters introduce, its covariance can become overly optimistic. This is a known problem called *filter smugness*. Unmodeled errors during the filtering procedure can have a big impact in the filter performance and even lead to divergence of the solution. In space navigation, there are several parameters (e.g., drag coefficients, SPR area, sensor, and actuators biases), both in the dynamics and the measurements models, which can be considered as constant (but unknown) in the OD process.

One of the most common algorithmic solutions to this problem was proposed by Schmidt [10]. Schmidt's approach was to perform the update step of a subset of state (*solve-for* states) while maintaining the covariance of all the states, including the states not involved in the update step (*consider* states). In this configuration, only the *solve-for* states are estimated. This kind of filter is also known as reduced-order filter, and it has been developed to reduce the computational effort of navigation filters estimating only a subset of the state. Consider filtering is a very efficient way of evaluating the effect of the uncertainty of some parameters in the estimation solution without having to extend the state vector variable and, therefore, reducing considerably the number of operations to be performed during the filtering process.

The general formulation considering a discrete linear system is derived in this paragraph. Let's consider a linear discrete-time system (without control action for sake of simplicity):

$$\mathbf{x}_k = \mathbf{F}_{k-1} \mathbf{x}_{k-1} + \mathbf{w}_{k-1}$$

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k.$$

This linear system can be rewritten considering *solve-for* states $\bar{\mathbf{x}}_k$ and *consider* state $\bar{\bar{\mathbf{x}}}_k$ as:

$$\begin{bmatrix} \bar{\mathbf{x}}_k \\ \bar{\bar{\mathbf{x}}}_k \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{F}}_{k-1} & 0 \\ 0 & \bar{\bar{\mathbf{F}}}_{k-1} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_{k-1} \\ \bar{\bar{\mathbf{x}}}_{k-1} \end{bmatrix} + \begin{bmatrix} \bar{\mathbf{w}}_{k-1} \\ \bar{\bar{\mathbf{w}}}_{k-1} \end{bmatrix}$$

$$\mathbf{y}_k = \begin{bmatrix} \bar{\mathbf{H}}_k & \bar{\bar{\mathbf{H}}}_k \end{bmatrix} \begin{bmatrix} \bar{\mathbf{x}}_k \\ \bar{\bar{\mathbf{x}}}_k \end{bmatrix} + \mathbf{v}_k.$$

Note how, even if the *consider* state is not estimated, its effect is taken into account in the equations by “polluting” the estimation of $\bar{\mathbf{x}}_k$.

Similarly, the estimation error covariance \mathbf{P}_k can be partitioned as follows:

$$\mathbf{P}_k = \begin{bmatrix} \bar{\mathbf{P}}_k & \Sigma \\ \Sigma^T & \bar{\bar{\mathbf{P}}}_k \end{bmatrix}$$

And the resulting Kalman gain for the *solve-for* states takes the expression:

$$\bar{\mathbf{K}}_k = \left(\bar{\mathbf{P}}_k^{-} \bar{\mathbf{H}}_k^T + \Sigma_k^{-} \bar{\bar{\mathbf{H}}}_k^T \right) \alpha_k^{-1}.$$

where α_k is defined as:

$$\alpha_k = \bar{\mathbf{H}}_k \bar{\mathbf{P}}_k^- \bar{\mathbf{H}}_k^T + \bar{\mathbf{H}}_k \Sigma_k^- \bar{\bar{\mathbf{H}}}^T_k + \bar{\bar{\mathbf{H}}}_k (\Sigma_k^-)^T \bar{\mathbf{H}}_k^T + \bar{\bar{\mathbf{H}}}_k \bar{\bar{\mathbf{P}}}_k^- \bar{\bar{\mathbf{H}}}^T_k + \mathbf{R}_k.$$

The Schmidt–Kalman filter formulations include a dynamical propagation:

$$\hat{\mathbf{x}}_k^- = \bar{\mathbf{F}}_{k-1} \hat{\mathbf{x}}_{k-1}^+.$$

Similarly, for the covariance:

$$\bar{\mathbf{P}}_k^- = \bar{\mathbf{F}}_{k-1} \bar{\mathbf{P}}_{k-1}^+ \bar{\mathbf{F}}_{k-1}^T + \bar{\mathbf{Q}}_{k-1}$$

$$\bar{\bar{\mathbf{P}}}_k^- = \bar{\bar{\mathbf{F}}}_{k-1} \bar{\bar{\mathbf{P}}}_{k-1}^+ \bar{\bar{\mathbf{F}}}_{k-1}^T + \bar{\bar{\mathbf{Q}}}_{k-1}$$

$$\Sigma_k^- = \bar{\mathbf{F}}_{k-1} \Sigma_{k-1}^+ \bar{\mathbf{F}}_{k-1}^T.$$

The best estimate for the *solve-for* states is then obtained by performing the *state and covariance update* step and it can be expressed as:

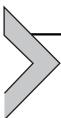
$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \bar{\mathbf{K}}_k (\mathbf{y}_k - \bar{\mathbf{H}}_k \hat{\mathbf{x}}_k^-)$$

$$\bar{\mathbf{P}}_k^+ = (\mathbf{I} - \bar{\mathbf{K}}_k \bar{\mathbf{H}}_k) \bar{\mathbf{P}}_{k-1}^- - \bar{\mathbf{K}}_k \bar{\bar{\mathbf{H}}}_k (\Sigma_k^-)^T$$

$$\Sigma_k^+ = (\mathbf{I} - \bar{\mathbf{K}}_k \bar{\mathbf{H}}_k) \Sigma_k^- - \bar{\mathbf{K}}_k \bar{\bar{\mathbf{H}}}_k \bar{\bar{\mathbf{P}}}_k^-$$

$$\bar{\bar{\mathbf{P}}}_k^+ = \bar{\bar{\mathbf{P}}}_k^-.$$

Summarizing, Schmidt–Kalman filters can be used to avoid filter smugness in presence of unknown parameters in the dynamical or observation model. This is done by updating only a subset of state (*solve-for* states) while maintaining the covariance of all the states, including the parameters that are not estimated. In this way, the effect of the uncertainty of these parameters is taken into account during the estimation procedure without augmenting the state vector, reducing the overall computational effort.



Batch estimation

In contrast to sequential filtering techniques presented in the previous sections, batch filters cover a different set of filtering techniques that will be explained in this chapter.

Batch filters are typically used when performing trajectory reconstruction, thus postflight analysis in which all the information and available

measurements are taken into account and fused, in order to properly determine the trajectory, a particular spacecraft (S/C) has followed during a given period of time. This process is usually carried out on-ground since the computational load is too high to be performed on-board.

Least squares

The least squares solution is the classical method used to find the best fit of a given set of data points by minimizing the sum of the squares of the computed residuals with respect to the estimated values. In space navigation, the target is to minimize the measurement residuals, thus the differences between the real measurements coming into the navigation filter from the different sensors in the S/C and the estimated values of those measurements are generated using the internal models by navigation.

The estimated measurements are generally represented as:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{v}$$

where \mathbf{y} is the measurement value(s), \mathbf{x} is the vector of length n including navigation variables (typically S/C position, velocity, mass, and any extra variable/parameter included in the navigation process, as mentioned in the section above), and \mathbf{H} represents the measurement model: the different equations that relate the navigation variables in \mathbf{x} and combine them with other known parameters to generate the estimated measurement. The vector \mathbf{v} refers to the observation error. Therefore, the idea behind the least squares is to find the \mathbf{x} that minimizes the following:

$$J(\mathbf{x}) = \frac{1}{2} \mathbf{v}^T \mathbf{v} = \frac{1}{2} \sum_{i=1}^m \mathbf{v}_i^T \mathbf{v}_i = \frac{1}{2} (\mathbf{y} - \mathbf{H}\mathbf{x})^T (\mathbf{y} - \mathbf{H}\mathbf{x})$$

with m being the number of measurements. In order to find the minima for the quadratic function of \mathbf{x} expressed above, two conditions must be fulfilled, the first derivative must be equal to zero and the second derivative needs to be positive:

$$\frac{\delta J}{\delta \mathbf{x}} = 0$$

$$\delta \mathbf{x}^T \frac{\delta^2 J}{\delta \mathbf{x}^2} \delta \mathbf{x} = \delta \mathbf{x}^T \mathbf{H}^T \mathbf{H} \delta \mathbf{x} > 0$$

for any $\delta \mathbf{x} \neq 0$.

Solution of the first equation leads to Ref. [11]:

$$(\mathbf{H}^T \mathbf{H}) \hat{\mathbf{x}} = \mathbf{H}^T \mathbf{y}$$

Being $\mathbf{H}^T \mathbf{H}$ an $n \times n$ matrix, if it is full rank (rank n), it will be positive definite and the best estimate for \mathbf{x} is:

$$\hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{y} = \mathbf{H}^{-1} \mathbf{y}$$

In case there are more equations (m) than variables (n), it means that \mathbf{H} is an $m \times n$ matrix and, therefore, $\mathbf{H}^T \mathbf{H}$ an $m \times m$ matrix. In this case, there are infinite solutions for \mathbf{x} , and the minimum norm solution can be selected:

$$\hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{y}$$

where $(\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T$ is the so-called pseudoinverse of \mathbf{H} .

Dynamic effects

During the estimation process in space environment, the dynamical effects also need to be taken into account. In the formulation presented in the previous section, the least squares solution allows determining the best \mathbf{x} estimation at a given epoch in terms of minimizing the sum of the squares of the residuals. However, the dynamics of the S/C in orbit cannot be neglected, since typically the measurements are not always referred to the same epoch but distributed along a trajectory arc. Therefore, each measurement \mathbf{y}_k is related to a different value of the state vector at time k :

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

This means that, given m different measurements distributed along a trajectory arc, each one referred to a different epoch, during the estimation process, m different state vectors are defined, since the position and velocity of the S/C evolve along time according to the dynamics of the orbit. Therefore, it is required to express all the measurements related to the same point of the orbit, so all of them referred to the same state vector. This allows using the least squares formulation to estimate \mathbf{x} at a particular point in the orbit, which typically is the starting epoch of the considered trajectory arc. In order to perform this step, the state transition matrix (STM) is employed.

The dynamics of a spacecraft in orbit around a given body, including perturbation effects, follows nonlinear relationships:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t) \mathbf{x}(t)$$

with $\mathbf{x}(t_0) = \mathbf{x}_0$. If a reference trajectory reasonably close to the true one is available, it is possible to linearize the problem. The general solution of this linearized system can be expressed via the STM:

$$\mathbf{x}(t) = \Phi(t, t_i)\mathbf{x}(t_i)$$

where $\Phi(t, t_i)$ is the STM relating the states at a given epoch t_i with the states at t . Since the value at t_0 is available, every point can be related to this one:

$$\mathbf{x}(t) = \Phi(t, t_0)\mathbf{x}(t_0) = \Phi(t, t_0)\mathbf{x}_0$$

Being the STM:

$$\Phi(t, t_0) = \frac{\partial \mathbf{x}}{\partial \mathbf{x}_0}$$

Therefore, the STM can be then used to link the observations to the same epoch, t_0 , in order to apply the least squares algorithm and obtain the estimation of the state vector at the beginning of the considered trajectory arc:

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \mathbf{z}_k = \mathbf{H}_k \Phi(t, t_0) \mathbf{x}(t_0) + \mathbf{v}_k$$

So, given a trajectory arc, the steps to perform the estimation of the state vector are as follows:

- Calculation of the STMs between measurement epochs
- Generation of estimated measurement according to the measurement models presented in the navigation algorithms
- Reference all the measurements to the initial epoch of the arc using the STMs, so all of them are related to $\mathbf{x}(t_0)$
- Estimate the state vector at t_0 using the least squares methods to obtain $\hat{\mathbf{x}}(t_0)$
- Propagation of $\hat{\mathbf{x}}(t_0)$ from t_0 to the final time of the trajectory arc t_f to obtain the solution of the state vector (and other navigation variables) along it.

This last propagation can be performed either using the STMs or a numerical propagator. In strongly nonlinear environments, using the STM could reduce the accuracy of the solution at the end of the arc.

Effect of observation errors

The least squares method presented up to now does not take into account any observation errors or a priori information of the parameters to be estimated. Up to this point, it has been assumed that all measurements have the same level of uncertainty. However, in the most general case, some

measurements will probably be more reliable than others. Therefore, the observations with higher level of confidence should be given a higher weight or preference during the estimation process to obtain the best possible estimate with the available information. Typically, the confidence level of the measurements is given by the observation error: lower observation errors imply higher confidence in measurements and vice versa. Regarding the effect of the observation errors in the estimation, the problem is stated as follows:

$$\mathbf{y}_k = \mathbf{H}_k \cdot \mathbf{x}_k + \mathbf{v}_k$$

$$E[\mathbf{v}\mathbf{v}^T] = \mathbf{R} = \begin{bmatrix} R_{11} & \cdots & R_{1m} \\ \vdots & \ddots & \vdots \\ R_{m1} & \cdots & R_{mm} \end{bmatrix}$$

Usually, the observation errors are not correlated, therefore $R_{ij} = 0$ for $i \neq j$, but this is not required in the formulation and in the general case it can be assumed that $R_{ij} \neq 0$. Therefore, the cost function to be minimized is then:

$$J(\mathbf{x}) = \frac{1}{2} \mathbf{v}^T \mathbf{v} = \mathbf{v}^T \mathbf{R}^{-1} \mathbf{v}$$

By computing the first derivative with respect to \mathbf{x} and equating it to zero to obtain the best estimate of \mathbf{x} :

$$\hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{y}$$

It is clear from the previous equation that in order for this method to work, the matrix \mathbf{R} cannot be singular, so observations cannot be perfect. Luckily, real-world observations have always some errors associated, so the user needs to take care of relating a nonnull error to every measurement in the formulation. The associated estimation error covariance is also affected by the observation errors in the following way:

$$\mathbf{P} = E[(\hat{\mathbf{x}} - \mathbf{x})(\hat{\mathbf{x}} - \mathbf{x})^T] = (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1}$$

In order to propagate the covariance matrix, by looking at the definition, it can be seen that:

$$\begin{aligned} \mathbf{P}(t) &= E[(\hat{\mathbf{x}}(t) - \mathbf{x})(\hat{\mathbf{x}}(t) - \mathbf{x})^T] \\ &= E[\Phi(t, t_i)(\mathbf{x}(t_i) - \mathbf{x})(\mathbf{x}(t_i) - \mathbf{x})^T \Phi(t, t_i)^T] \end{aligned}$$

$$\mathbf{P}(t) = \Phi(t, t_i) \mathbf{P}(t_i) \Phi(t, t_i)^T$$

Inclusion of a priori information data

If a priori information is present, new observations can be used to update it. A simple graphical representation of this concept is shown in Fig. 9.8. This allows improving the estimation in a recursive way without the need of rerunning the algorithm from the beginning each time a new measurement is available. In trajectory reconstruction, this can be implemented by dividing the trajectory in arcs and using the solution of the previous arc as a priori information. So, assuming an a priori estimate and associated estimation error covariance is available:

$$\hat{\mathbf{x}}_0 = E(\mathbf{x})$$

$$\mathbf{P}_0 = E[(\mathbf{x} - \hat{\mathbf{x}}_0)(\mathbf{x} - \hat{\mathbf{x}}_0)^T]$$

The idea is to being able to update the estimation without the need of computing again all the past observations from the previous step to redo the calculations, just by adding a new batch of measurements taken since the last solution was obtained:

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_{k-1} + \mathbf{v}_k$$

$$\hat{\mathbf{x}}_k = \mathbf{x}_{k-1} + \mathbf{K}_k (\mathbf{y}_k - \mathbf{H}_k \mathbf{x}_{k-1})$$

So, the new estimation is obtained using the previous estimate and the new measurements. The matrix \mathbf{K}_k is the so-called estimator gain matrix and $(\mathbf{y}_k - \mathbf{H}_k \mathbf{x}_{k-1})$ is the measurement residual used to correct the a priori information with the observations of the new arc. The value of \mathbf{K}_k is chosen to minimize the sum of the variances of the estimation errors at k :

$$J_k = E[(\mathbf{x}_1 - \hat{\mathbf{x}}_1)^2] + E[(\mathbf{x}_2 - \hat{\mathbf{x}}_2)^2] + \cdots + E[(\mathbf{x}_n - \hat{\mathbf{x}}_n)^2] = Tr(\mathbf{P}_k)$$

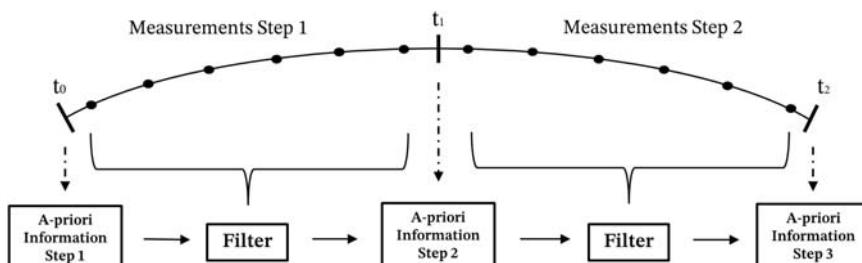


Figure 9.8 Inclusion of a priori information.

By differentiating the previous equation and setting it to zero, the value of \mathbf{K}_k that minimizes J_k is obtained as:

$$\mathbf{K}_k = \mathbf{P}_{k-1} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_{k-1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$

Next to the update of the state, the covariance of the estimation error can also be updated recursively with the equation (to check the mathematical formulation that leads to it, the reader is invited to check Ref. [5]):

$$\mathbf{P}_k = (\mathbf{I} + \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k-1} (\mathbf{I} + \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T$$

where \mathbf{R}_k is the covariance matrix of the observations as defined above. Therefore, according to the above, starting from an a priori estimation and its associated covariance, the new estimate can be obtained recursively by computing first the \mathbf{K}_k matrix and using it to correct the previous values of $\hat{\mathbf{x}}_k$ and \mathbf{P}_k .

Problems in batch orbit determination

Different issues may arise during the estimation process. A hint of some of these problems is presented in this section.

Presence of nonlinearities

The techniques presented up to this point refer to linear systems. However, real systems are nonlinear by nature, linear systems do not really exist. Therefore, even if these techniques can still be applied to real systems, they are limited by how much the studied system can be approximated by a linear one. If a particular system is close enough to linear, linear estimation techniques will approach well and the estimation results will be valid.

If the effect of nonlinearities is nonnegligible, extra errors can be injected into the process due to the use of the STM to propagate the covariance matrix, the state, and also to refer the different measurements at different times to the same epoch. In order to overcome and/or mitigate these effects, a numerical propagator can be used to propagate the states after the update instead of using the STM. In this way, the nonlinearities of the dynamics will be maintained by the propagator if the STM is not able to do so.

However, for the situations in which the use of the STM is mandatory, like the propagation of the covariance or referring the measurements to the same epoch, the alternative could be reducing the length of the arc to which the batch is applied, in order to reduce the effect of the nonlinearities that cannot be captured by the STM.

Incorrect a priori statistics and unmodeled parameters

The use of incorrect a priori statistics to initialize the filter leads to convergence problems in some cases since the uncertainty level of the initial estimate does not correspond to reality. This means that, if the associated covariance of the estimation error shows that the confidence level of the solution is high, but, in reality, the estimate is much further from the real solution, the filtering process will tend to trust the a priori solution more than it should. This can reach a point in which good observables are discarded since the filter assumes that its current solution has a confidence level higher than the one of the new measurements. This level of confidence of the a priori solution of the estimate is hard to know since the “real world” is not known (that is why navigation filters are needed). Therefore, a detailed analysis is required to assess the level of error of the a priori solution.

Apart from this, the effect of unmodeled parameters also has a big impact in the convergence of the filtering process. If the estimated world modeled by navigation does not take into account a major contributor in the evolution of the spacecraft dynamics, the filter will not be able to accurately track the dynamics internally, and there will be a discrepancy between the evolution of the real position and the estimated one.

It is important to mention here that the fact of ignoring the effect of a parameter has more impact than knowing it exist and associate a high uncertainty to it. For example, knowing that a parameter has an effect on the dynamics (e.g., solar radiation pressure - SRP) but, if due to the available data, it is not possible to model it correctly, can be mitigated by associating a higher uncertainty to it. This would result in higher uncertainty in the estimate solution, but it will not likely lead to a divergence in the filtering process. However, ignoring the fact that SRP is present in the dynamics would make the filter to trust too much its internal propagation, since the associated uncertainty would be very low. In this latter case, the filter would converge to a wrong solution, but without knowing it is wrong, which could lead to divergence in the navigation.

Numerical problems

Due to the finite precision of computers, numerical problems may arise during the filtering process, especially related to the covariance properties. Computational errors can lead to the loss of symmetry of the covariance matrix or its positive definite nature. This kind of errors could produce the divergence of the filtering process.

In order to prevent and mitigate the limited precision of the calculations, alternative filtering algorithms can be used. These are related to different ways of decomposing and factorizing the covariance matrix, helping toward the robustness of the algorithm by keeping the characteristics of the covariance matrix throughout the whole filtering process.

Square root information filter

Square root filters present improved numerical precision and stability, which are very useful for bad-conditioned problems. The idea behind square root filters is to replace the covariance matrix and use its square root instead during the recursive process of OD. In this way, the algorithm will assure the positive semidefiniteness of the error covariance. So instead of \mathbf{P}_k , the update and propagation steps explained above will be performed over the matrix \mathbf{S}_k :

$$\mathbf{P}_k = \sqrt{\mathbf{P}_k} \sqrt{\mathbf{P}_k}^T = \mathbf{S}_k \mathbf{S}_k^T$$

Since the covariance square root is not uniquely defined, the common practice is to use a specific formulation of \mathbf{S}_k that is attractive for the algorithm implementation, such as upper or lower triangular matrices. This reduces the amount of computation and storage when implementing the algorithm.

If the square root decomposition is applied to the inverse of the covariance matrix, the information matrix, the resulting algorithm is the so-called Square Root Information Filter (SRIF), widely used in orbit reconstruction procedures due to its numerical precision and stability:

$$\mathbf{P}_k^{-1} = \mathbf{S}_k^{-T} \mathbf{S}_k^{-1}$$

The algorithms for the SRIF exploit the properties of orthogonal transformations to perform the update and the propagation of the state and square root information matrix. The filter mechanization works with \mathbf{z} and \mathbf{S} , with \mathbf{z} related to \mathbf{x} as detailed below:

$$\mathbf{x} = \mathbf{S}^{-1} \mathbf{z}$$

The update step of the filtering process with additional observation(s) is performed by applying a series of orthogonal transformations \mathbf{T} (for more details on the orthogonal transformations, the reader is invited to check [11,12]):

$$\mathbf{T} \begin{bmatrix} \mathbf{S}_k & \mathbf{z}_k \\ \mathbf{H}_k & \mathbf{y}_k \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{S}}_k & \hat{\mathbf{z}}_k \\ 0 & \mathbf{e}_k \end{bmatrix}$$

With \mathbf{e}_k the observation residual, $\hat{\cdot}$ representing the updated variables and:

$$\hat{\mathbf{S}}_k \hat{\mathbf{x}}_k = \hat{\mathbf{z}}_k$$

$$\mathbf{P}_k = \hat{\mathbf{S}}_k^{-1} \hat{\mathbf{S}}_k^{-T}$$

Then the propagation step is performed by making use of the STM:

$$\mathbf{S}_k = \mathbf{S}_j \boldsymbol{\Phi}^{-1}(t_k, t_j)$$

$$\mathbf{z}_k = \mathbf{S}_k \boldsymbol{\Phi}^{-1}(t_k, t_j) \hat{\mathbf{x}}_j$$

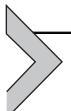
U-D filter

In alternative to the SRIF formulation, the U-D covariance factorization expresses the covariance matrix as:

$$\mathbf{P}_k = \mathbf{U}_k \mathbf{D}_k \mathbf{U}_k^T$$

with \mathbf{U}_k an upper triangular matrix and \mathbf{D}_k a diagonal one. The U-D filter shares the properties of the SRIF filter in terms of numerical stability and accuracy and the fact that assures the positiveness of the covariance. On top of that, since no square root operations are necessary, it is computationally more efficient although this affirmation strictly depends on the number of state variables and measurements: when the number of measurements is very large, the difference in performance is considerably reduced.

Please note that the derivation of the UD algorithms can be done by including the definition of \mathbf{P}_k as upper triangular and diagonal matrices into the least squares equations and making use of the characteristics of these matrices. For a complete derivation, the reader is suggested to check [13].



Absolute orbit navigation

Absolute orbit navigation functions estimate the translational orbital states of the spacecraft, in order to know the current spacecraft position and velocity, and possibly propagate the future states by exploiting the orbital dynamics laws.

GNSS spacecraft navigation

Absolute orbit navigation often exploits the GNSS constellations to estimate the orbital state of the spacecraft. The main concepts of GNSS sensors and their measurements are introduced in [Chapter 6 - Sensors](#). GNSS refers to a constellation of satellites around the Earth providing signals and, therefore, position and velocity information to a “user” receiving them. In this section, GNSS observables and how to use them inside a navigation filter are detailed.

GNSS observables

The GNSS observables are the pseudorange, the carrier phase, and the Doppler measurements, and they are detailed in the following sections.

Pseudorange

The pseudorange represents the distance between the GNSS satellite at the time of the emission of the signal and the receiver antenna measured at the time of reception. Assuming no errors, the pseudorange $\tilde{\rho}$ would be equal to the geometric distance:

$$\tilde{\rho} = c(t_R - t_T) = \rho(t_R, t_T)$$

Being c the speed of light, t_R the time of the clock receiver, and t_T the time of the transmitter clock. However, in practice, the pseudorange is not equal to the geometrical distance due to different factors, such as the errors in the clocks, the effect of the atmosphere, Earth tides, multipath effects, and relativistic effects. Therefore, the real pseudorange observable is given by the expression:

$$\tilde{\rho} = \rho(t_R, t_T) - c(\delta t_R - \delta t_T) + \delta_{iono} + \delta_{tropo} + \delta_{tide} + \delta_{path} + \delta_{rel} + \epsilon$$

where δt represents the clock errors, δ_{iono} and δ_{tropo} the effect of the ionosphere and troposphere in the signal transmission, δ_{tide} the errors associated to the Earth tides, δ_{path} the multipath effects, and δ_{rel} the relativistic effects. The other nonmodeled errors are represented by ϵ . All the effects in the previous equation are briefly explained in a dedicated section below.

Carrier phase

The carrier phase measures the phase of the received satellite signal with respect to the carrier phase generated in the receiver at the reception time. It represents a measurement of the distance between emitter and receptor given in cycle units of the carrier frequency. So, the observable is

obtained by shifting the generated signal carrier phase to match it with the received carrier phase from the satellite. Carrier phase measurements are generally much more precise than pseudorange measurements, since it can be tracked with a precision in the order of the millimeters. However, there are two key points that must be taken into account:

- *Phase ambiguity.* Adding an arbitrary constant integer number of cycles to the transmitted carrier signal would result in the same measured phase. Different methods based on conditional adjustment theory have been developed in the last decades to solve the phase ambiguity problem. For a detail description of such methods, the reader is invited to check Chapter 8 of Ref. [14].
- *Cycle slip.* Phase must be tracked continuously since a gap in the tracking of the phase over time would change the value of the phase ambiguity. Assuming no errors and vacuum medium for the transmission, the measured phase Φ follows the expression:

$$\Phi = \Phi_R(t_R) - \Phi_T(t_R) + N_R^T$$

where Φ_R is the phase of the receiver, Φ_T is the phase of the received signal from the satellite, and N_R^T is the ambiguity between satellite and receiver. Taking into account the relationship between the speed of light c , wavelength λ , and frequency f : $c = \lambda f$; and accounting for all the errors as in the pseudorange case, the carrier phase measurement model is as follows:

$$\begin{aligned} \lambda\Phi = & \rho(t_R, t_T) - c(\delta t_R - \delta t_T) + \lambda N_R^T - \delta_{iono} + \delta_{tropo} + \delta_{tide} \\ & + \delta_{path} + \delta_{rel} + \varepsilon \end{aligned}$$

The negative sign of the ionosphere effect is due to the fact that the ionosphere advances the phase signal transmission, whereas, in case of the pseudorange signal, the ionospheric effect delays it (so the sign in the pseudorange equation above is positive). For a more detailed explanation on the ionospheric effects in pseudorange and phase measurements, the reader is invited to check Ref. [14].

Doppler measurements

Doppler measurements are based on the physical phenomenon that shifts the frequency of a signal due to the fact that the emitter moves toward or away from the receiver. The Doppler shift can be used as an independent observable to provide direct observation of the instantaneous range rate in the

direction defined by the vector emitter—receptor. Since the original frequency of the emitter is replicated in the GNSS receiver, the shift between this last one and the real received signal can be measured. Of course, the errors in clocks need to be accounted for in the measurement equation, as expressed below.

The observable model of the Doppler shift is given by Ref. [14]:

$$D = \frac{d\rho(t_R, t_T)}{\lambda dt} - f \frac{d(\delta t_R - \delta t_T)}{dt} + \delta_f + \epsilon$$

where δ_f is the frequency correction due to relativistic effects. The atmosphere effects have no contribution to errors in the case of Doppler measurements.

Error effects

In this section, a brief summary of the errors inducted in the pseudorange, phase, and Doppler equations defined before is presented.

Ionospheric effects

As mentioned before, the ionosphere affects differently the pseudorange and the phase measurements. Whereas the pseudorange is delayed by the free electrons in the ionosphere, the phase is advanced. This different behavior is modeled by using different signs when including the ionosphere in the pseudorange and phase equations, respectively.

The effect of the ionosphere is not constant in time, and its magnitude also depends on the frequency of the signals. It can be measured by the following expression [15]:

$$\delta_{iono} = \frac{40.3}{f^2} \cdot TEC$$

where TEC is the Total Electron Content in the zenith direction.

The ionospheric delay (or advance) effect can be eliminated by combining the GNSS phase measurements.

Tropospheric effects

The tropospheric effect, in opposition to the ionospheric one, does not depend on the frequency of the signals. The troposphere has a refraction effect on the signals which is dependent on the elevation: the lower the elevation, the bigger the path the signal needs to go through the troposphere and, therefore, the higher the effect.

The tropospheric effect has two components, the dry and the wet contributions. The dry effect is the one that contributes the most, about 80%–90% approximately; whereas the wet one related to the amount of water in the atmosphere accounts for the rest of tropospheric refraction.

Relativistic effects

Effects due to Einstein's General and Special Relativity theories need to be taken into account in the GNSS measurement equations due to the different velocities between the receiver and the emitter clock as well as the effect of gravity, since clocks on Earth surface are closer to a massive body than the ones on orbit around it. Furthermore, the eccentricity of the orbits as well as the Earth rotation have a nonnegligible effect on the GNSS observables.

Due to these effects, there is a mismatch between clocks in the order of several tens of microseconds which, translated to distances using the speed of light, implies that not correcting for relativity effects could add errors in positioning in the order of tens of km.

Earth tidal effects

These effects are consequence of the Earth deformation due to the gravity forces from the Sun and the Moon, and the effect of the ocean tide loads on the Earth surface. This deformation implies some displacements of the Earth surface that, in case GPSs, can be different among the different regions due to the variations in the Earth structure. For regional systems, as the ones mentioned in the beginning of this chapter, these effects could be neglected since the differences in displacements are not that important.

Multipath effects

Multipath effects are more related to the local environment of the receiver antenna since, due to the local geometry, it can be possible that the signal is received via different paths due to reflections. In order to mitigate the interferences associated to this effect, the design of the antennas to take into account side lobes geometry is important.

A nice overview of all the possible errors to be taken into account in the measurement model formulation or to be compensated while performing position estimation and their magnitude is collected in Ref. [16] and summarized here in [Table 9.1](#).

Table 9.1 Typical GNSS errors and uncertainty.

	Error	Magnitude	Uncertainty
Satellite	Center of mass position	—	2.5 cm
	Antenna phase center offset	0.5–3 m	10 cm
	Phase center variations	5–15 mm	0.2–1 mm
	Clock offset	<1 ms	2 cm
	Relativistic clock effects	10–20 m	
	Differential code biases	<15 ns	0.1–1 ns
Atmosphere	Troposphere	2.3 m	5 mm
	Ionosphere	<30 m	<1 m
Dynamics	Solid earth tide	<0.4 m	1 mm
	Ocean tides	1–10 cm	1–2 mm
	Pole tide	25 mm	—
	Atmospheric drag	<1.5 mm	—
Receiver	Phase center offset	5–15 cm	—
	Phase center variations	<3 cm	1–2 mm
Others	Phase wind-up	10 cm	—

GNSS navigation approaches

GNSS is widely adopted and well-established for terrestrial and airborne navigation. In this paragraph, instead, we will focus on how to apply the same concepts to spacecraft navigation. The main difference between Earth-based applications and spacecraft navigation is that, in general, the motion of a spacecraft can be predicted with good accuracy, while, on Earth, the motion of a generic GNSS user is often nondeterministic and prone to quick changes. For this reason, while pure kinematics approaches are used for Earth-based positioning algorithms (precise point positioning [PPP]), dynamical approaches are suggested while dealing with spacecraft. In fact, the knowledge of the dynamical system allows to reduce the overall number of estimation parameters and to propagate the spacecraft trajectory also when limited or no GNSS signals are available. Three main approaches are adopted and discussed here:

- *Precise orbit determination (POD)*. Ground-based method used to estimate the spacecraft position with the highest possible accuracy.
- *Real-time navigation*. The spacecraft position is estimated using real-time measurements on-board.
- *Relative GNSS navigation*. Two or more spacecraft in close proximity are considered and their GNSS measurements are fused together to improve

relative navigation accuracy. This process can be performed on-ground or on-board in real-time.

The details of each approach are discussed in the following paragraph, but a first distinction can be done in terms of achievable performance. Fig. 9.9 shows the achievable accuracy for each of the GNSS navigation approaches.

From Fig. 9.9, the advantage of having an on-ground processing is evident as well as the fact that differential GNSS processing offers a high-accuracy solution. In this paragraph, an introduction to the most common positioning method is presented, and then, the peculiarities of spacecraft navigation are detailed.

Precise point positioning

PPP is one of the most common techniques for GNSS-based positioning. This method, introduced in 1997 [17], uses dual-frequency, pseudorange, and carrier-phase observations along with precise satellite orbit and clock to produce a PPP. PPP does not require simultaneous observations and exploits precise carrier-phase observations in addition to the pseudoranges. Most of the error sources listed in the previous paragraph are taken into account, i.e., carrier-phase ambiguities, tropospheric propagation delay, Earth tides, ocean tides, satellite and receiver antenna offsets, and carrier-phase windup. Furthermore, carrier-phase is subject to cycle slip that has to be considered along with the previously mentioned effects, leading to slow PPP convergence. For ground-based applications, PPP can provide accuracies in the order of few centimeters (1-sigma) [16]. This method is mostly used for Earth-based application, and, therefore, it will not be detailed here. Additional information and an accurate description can be found in Ref. [16].

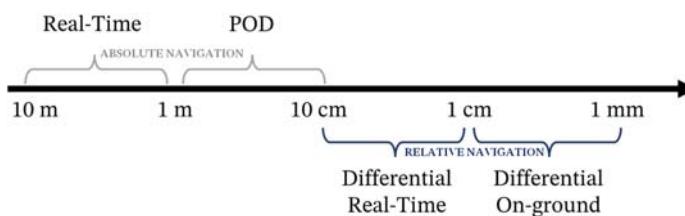


Figure 9.9 GNSS Navigation approaches and their accuracy.

Precise orbit determination

POD is a powerful method that allows to localize a spacecraft with high accuracy. Being a ground-based approach, it can exploit all the computational capabilities and processing resources to provide highly precise estimations. Nowadays, GNSS observations allow for OD of accuracy at maximum 1 cm 3D root-mean-square error (RMSE) for position [16]. The working principle of POD technique is similar to the one used for PPP but combining it with sophisticated spacecraft dynamics models. Also in this case, all the errors listed before must be considered and estimated. This includes, among others, relativistic clock and range corrections, phase center offsets and variations of the GNSS satellite and of the receiving antenna, phase delays and phase wind-up effects. It must be noted that the spacecraft dynamics describes the motion of the center of mass of the spacecraft while all the GNSS signals are referred to the antenna reference point. For this reason, the relative position between the GNSS antenna and the center of mass should be known with extreme accuracy. This is not always an easy task and internal perturbations (see [Chapter 3](#) – The Space Environment) should be modeled accurately. Furthermore, another important aspect is the time-tagging and synchronization of the measurements. Given the high velocities of a satellite orbiting the Earth, receiver and GNSS system time should be known with a high degree of precision. The most common approach to provide a POD solution is to use a batch method. Usually, a weighted least square estimator is employed to estimate all the dynamical parameters of the spacecraft (e.g., SRP area, drag coefficient, empirical accelerations) and GNSS observation model parameters (e.g., clock corrections, carrier phase ambiguities). Recalling the classical least square estimator formula:

$$J(\mathbf{x}) = \frac{1}{2} \mathbf{v}^T \mathbf{v} = \frac{1}{2} \sum_{i=1}^m \mathbf{v}_i^T \mathbf{v}_i = \frac{1}{2} (\mathbf{y} - \mathbf{Hx})^T (\mathbf{y} - \mathbf{Hx})$$

In this case, \mathbf{y} are the GNSS observables, \mathbf{H} a complex measurement model, and \mathbf{x} vector including all the navigation parameters. While designing \mathbf{x} , spacecraft and GNSS parameters should be considered. Usually, the following formulation is used:

$$\mathbf{x} = [\mathbf{T}, \mathbf{D}, \mathbf{B}]$$

with $\mathbf{T} = [cdt_1, \dots, cdt_n]^T$, vector containing the receiver clock offset parameters for n measurements, $\mathbf{D} = [\mathbf{r}_0, \mathbf{v}_0, C_D, C_R, a_{emp_1}, \dots, a_{emp_n}]^T$,

vector containing the initial position and velocity of the spacecraft, drag and SRP coefficients and empirical accelerations accounting for unmodeled effects, $\mathbf{B} = [b_1, \dots, b_n]^T$ vector containing the carrier-phase ambiguities for n measurement instants. The number of variables and parameters included in \mathbf{x} is an indicator of the high complexity of the measurement model in \mathbf{H} . Not having computational constraints, very long measurement vectors can be considered (one or more days) increasing the final estimation accuracy.

Real-time navigation

GNSS signals can be used also to enhance the autonomous navigation capabilities of a spacecraft. In fact, they can be employed as additional measurements to perform on-board spacecraft navigation, combined with a sequential filter. The working principle is very simple, and it is based on a classical sequential filter approach. In fact, a dynamical model is used to propagate the spacecraft state, and GNSS measurements are used to update the predicted spacecraft state. Usually, the dynamical model used on-board is much simpler than the one employed for POD, being the computational resources limited by the spacecraft avionics. A classical nonlinear EKF is often implemented to perform real-time navigation. Similar to POD solution, additional dynamical and clock biases can be added to the state vector and estimated inside the filter as parameters. For this reason, a Schmidt-EKF can be designed to deal with state and parameters estimation. Furthermore, an empirical acceleration term can be added to the classical gravitational (and nongravitational) accelerations to account for nonmodeled dynamical effects. The dynamical evolution of this term is not treated as a constant parameter, but it is usually modeled as exponentially correlated random variable:

$$a_{emp\ k} = e^{\frac{t_k - t_{k-1}}{\tau}} a_{emp\ k-1}$$

with τ being a “damping” tunable term.

For what concerns the observation model, two main approaches are usually employed:

- *GNSS-processed measurements.* The GNSS receiver processes the raw GNSS measurements providing position information to the KF. In this way, the measurement equation is very simple, and the filter and GNSS positioning algorithms are decoupled. No clock estimation is needed, and no GNSS position knowledge is required. However, with this approach, signal information is lost in the receiver kinematic

estimation and the precision of the solution is limited (around 5 m 3D RMSE for position – [16]). Furthermore, since the receiver kinematic positioning algorithm requires at least four GNSS signals to produce a valid measurement, real-time navigation is not possible when this condition is not satisfied.

- **GNSS raw measurements.** The GNSS raw measurements are directly used as measurements inside the KF. In this case, the observation model has to include the classical light-time corrected geometric relative distance formulation. Moreover, the GNSS satellite positions have to be known (e.g., through broadcast ephemeris) along with their clock biases. Ionospheric path delays should also be considered and corrected if ionosphere-free combinations [18] are not implemented. When dealing with carrier-phase measurements, it is important to estimate the ambiguities related to each GNSS satellite as additional parameters, highly increasing the computational cost and algorithmic complexity. The achievable results with this method are in the order of 0.5/1m 3D RMSE for position [16].

The selection of one of the two models strongly depends on the desired accuracy of the estimation algorithm and on the computational capabilities available on-board.

GNSS-INS integration GNSS measurements can also be used, in real-time, combined with an Inertial Navigation System (INS). An INS is a navigation system exploiting an IMU (see [Chapter 6 - Sensors](#)) to compute position, velocity, and attitude of a spacecraft. Even though INS short-term errors are small, they tend to drift in time, accumulating errors and quickly degrading the measurement accuracy ([Chapter 6 - Sensors](#)). For this reason, GNSS and INS are often used together to take advantage of the long-term stability typical of GNSS systems. In the other hand, INS can also compensate for partial unobservability of GNSS satellites, propagating the spacecraft state without having to rely on GNSS measurements. Depending on the type of GNSS measurements processing (i.e., GNSS-processed measurements or GNSS raw measurements), the GNSS-INS integration architecture is called loosely or tightly coupled. The considerations drawn before for the two architectures, in terms of precision, computational complexity and model complexity are also valid for GNSS-INS integration. For the sake of clarity, the high-level schematic of the two approaches is reported in [Fig. 9.10](#).

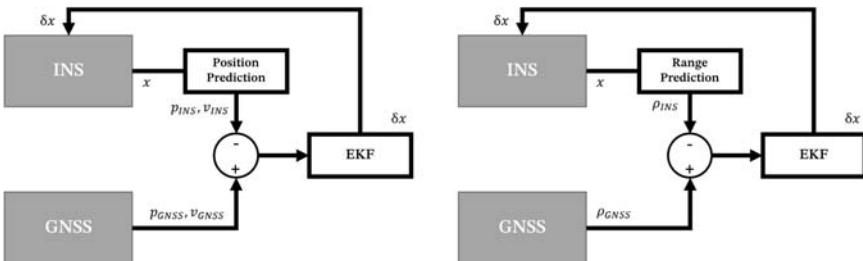


Figure 9.10 GNSS-INS architecture: loosely coupled (left) and tightly coupled (right).

In the loosely coupled architecture, the GNSS receiver provides directly position and velocity measurements. The difference between GNSS and INS position and velocity measurements are used inside an EKF to estimate the error in the state vector δx which is fed to the INS block to remove eventual drifts. On the contrary, for a tightly coupled architecture, the GNSS provides the raw measurements (pseudorange in this case) that are compared to the one measured by the INS to compute the state vector error, thanks to an EKF. The estimated state error is used to compensate the INS long-term errors.

Relative GNSS navigation

The benefit of having two or more spacecraft orbiting in close proximity is that differential GNSS techniques can be employed. In fact, several common sources of error can be canceled out when processing differential measurements. This is the case of ionospheric path delays, GNSS orbits, and clock errors. Also, carrier-phase ambiguities are easier to be resolved for differential GNSS measurements. Cm-level accuracy can be achieved exploiting this kind of algorithms. An important aspect to consider while implementing differential GNSS algorithms is the synchronization of GNSS measurements. In fact, proper synchronization is of vital importance to be able to correctly cancel all the common errors. Desynchronization would lead to highly degraded performance. Both on-ground POD and real-time navigation using nonlinear KFs are possible with differential GNSS. The implementation details resemble the ones discussed throughout the section. Further details can be found in Ref. [16].

Pulsar-based spacecraft navigation

The estimation of the spacecraft state (position and velocity) with respect to an inertial frame is a difficult task, especially for missions far from low Earth

orbit where no GNSS signals are available. In these cases, the navigation solution typically needs to rely on radiometric techniques that provide a direct range and/or range rate measurement in the LOS direction. Radiometrics are heavily affected by delays, particularly for exploration missions to outer planets, asteroids, or comets, and, on top of that, the cost of using the ground stations of the Deep Space Network is considerably high taking into account the high demand for its services. In order to mitigate these problems, it is possible to use pulsar measurements to obtain a navigation solution with respect to an inertial reference frame in the Solar System. This would help not only to improve the spacecraft state knowledge but also to increase the autonomy level for exploration missions that are limited by the round-trip communications delay with Earth.

Pulsars are neutron stars in a rapid rotation state generating electromagnetic pulses due to the conservation of their angular momentum during the collapse of the star. Young pulsars tend to rotate with very high rotation periods in the order of milliseconds, whereas the rotation of the older ones is of several seconds. The main characteristic of pulsars is the great stability and predictability of their rotation period. By analyzing the orientation, pulse frequency, and pulse shape, the received signals can be uniquely associated to a known pulsar in a catalog. The algorithm is based on Time of Arrival (TOA) observables: the received signal is compared to an internal catalog that contains the predicted one at the Solar System Barycenter (SSB). The differences in arrival time between the predicted at the SSB and the actual one on-board can be easily translated to a range distance from SSB to spacecraft in the direction of the pulsar by multiplying it by the speed of light. This also required the knowledge of the pulsar coordinates in a known reference frame (like ICRF) to determine the direction in which the range measurement is projected. Since pulsars are very far from the Solar System, it can be assumed that their direction seen from the SSB is the same as the one observed from the spacecraft.

Therefore, in theory, using only a given and already known catalog onboard to compare the arrival signals, the spacecraft can autonomously compute its state with respect to the SSB, especially if different pulsars are observed, which would provide range measurements along different directions ([Fig. 9.11](#)).

The navigation measurement model is then relying on an a priori state solution, generally coming from an internal propagator, and some extra pulsar known characteristics:

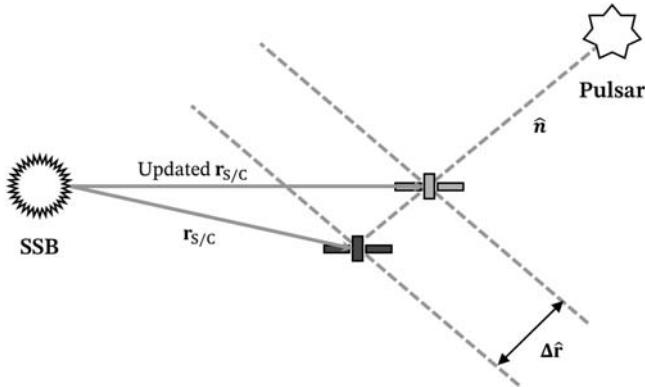


Figure 9.11 Pulsar-Based navigation scheme.

$$range_{SSB-S/C} = c \left(t_{SSB} - t_{S/C} \right) = f \left(\mathbf{r}_{S/C}, \hat{\mathbf{n}}, \mathbf{D}_P, \mathbf{V}_P \right)$$

where $\mathbf{r}_{S/C}$ is the spacecraft position vector, $\hat{\mathbf{n}}$ is the unit vector representing the direction of the pulsar with respect to the SSB, and \mathbf{D}_P and \mathbf{V}_P are the known position and proper motion of the pulsar with respect to the SSB. Even if the effect of the proper motion is very small compared to the other terms in the equations, it can be included for high-precision modeling of the pulsar measurement. The extended expression of the function is [19]:

$$\begin{aligned} f \left(\mathbf{r}_{S/C}, \hat{\mathbf{n}}, \mathbf{D}_P, \mathbf{V}_P \right) &= \hat{\mathbf{n}}_i \cdot \mathbf{r}_{SC} - \frac{r_{S/C}^2}{2\mathbf{D}_0} + \frac{(\hat{\mathbf{n}} \cdot \mathbf{r}_{S/C})^2}{2\mathbf{D}_0} + \frac{\mathbf{r}_{S/C} \cdot \mathbf{V}_P \Delta t_N}{\mathbf{D}_P} \\ &\quad - \frac{(\hat{\mathbf{n}} \cdot \mathbf{V}_P \Delta t_N)(\hat{\mathbf{n}} \cdot \mathbf{r}_{S/C})}{\mathbf{D}_P} - \frac{\mathbf{b} \cdot \mathbf{r}_{S/C}}{\mathbf{D}_P} \\ &\quad + \frac{(\hat{\mathbf{n}} \cdot \mathbf{b})(\hat{\mathbf{n}} \cdot \mathbf{r}_{S/C})}{\mathbf{D}_P} \\ &\quad + \frac{2\mu_{Sun}}{c^2} \ln \left[\frac{\hat{\mathbf{n}} \cdot \mathbf{r}_{S/C} + \mathbf{r}_{S/C}}{\hat{\mathbf{n}} \cdot \mathbf{b} + b} + 1 \right] \end{aligned}$$

Clock errors

The errors in the on-board clock will directly affect the navigation solution. Therefore, they need to be included in the navigation measurement model to track the behavior of the real clock. The lack of modeling of the clock parameter evolution can lead to very large errors that would be translated

to spacecraft position errors via the residuals in the filtering process. The clock errors affecting the measurements consist of three main contributions:

- *Clock bias.* This can be considered as a constant error present during a complete simulation. This error affects all the received measurement in the same way, leading to a TOA bias that will translate into a spacecraft distance bias with respect to the SSB. The effect of the clock bias can be mitigated and reduced to the minimum via ground calibration.
- *Clock drift.* These errors are related to the short-term instabilities of the clock. They are not random, and its characteristics time scales are in the same order of the duration of a single observation. Therefore, they will change between measurements and cannot be considered constant, and for this reason, it is mandatory to model them during the navigation process. Clock drifts evolution can be modeled using a quadratic expression with time in which the three coefficients need to be estimated. The estimation of these coefficients would allow tracking the behavior of the lock errors, keeping them from polluting the navigation solution.
- *Clock jitter.* These are high-frequency random errors with lower order of magnitude than the bias and drift. As the previous ones, these errors will affect the pulsar measurements, but they cannot be modeled due to their random nature. They are taken into account into the navigation procedure as measurement noise.

The need to estimate the extra parameters of the clock implies that extra measurements are required for the system to be completely observable. Typically, at least four different measurements are needed.

Ephemerides error

Finally, there is an extra aspect to be taken into account with pulsar navigation: the effect of the ephemerides error. As mentioned before, pulsar observations provide a range measurement from the SSB to the spacecraft by comparing the received signal on-board with the expected one in the SSB. However, if the spacecraft is orbiting a body with limited ephemerides knowledge, they must be taken into account into the filtering process. Typically, navigation filters predict the spacecraft position with respect to a central body. Therefore, an error in the position of that central body with respect to the SSB needs to be accounted for into the measurement model. Missing this point would imply that the ephemerides error is translated to spacecraft relative position error via the residuals in the navigation filter.

Ground-based orbit determination

OD is the process by which tracking data is used to compute the spacecraft's orbital states. It is inherently a navigation process, but since it is performed on-ground and not on-board, the common terminology refers to it as "ground-based OD." In absolute OD, the spacecraft's position and velocity are computed relative to a Solar System body in an absolute sense, rather than relative to other moving spacecraft. The body for which the orbit is referred to depends on the type and phase of the mission. For example, the trajectory of a spacecraft traveling to Mars will first be determined relative to Earth for launch and initial acquisition. After it leaves the Earth's vicinity, the orbit will be computed relative to the Sun or the barycenter of the Solar System, and finally, as the spacecraft enters the vicinity of Mars, the orbit will be computed relative to Mars itself.

The OD process itself is fundamentally a nonlinear least squares problem. First, there are the nonlinear equations of motion that describe the spacecraft's orbit. Then, there are observational data which have a nonlinear relationship between its value and the orbit. The goal is to adjust the orbit (and parameters associated with the orbit and the data) in a least squares sense to minimize the difference between the predicted and actual value of the observations (called the residuals). The method typically used in the ground OD process is the batch filter, as described earlier, linearizing around a pre-determined reference trajectory. The use of the batch filter is preferred, using a factorization method like UD, due to its superior ability to easily remove outlier data points.

The data types used for OD vary depending on the type of mission. For satellites orbiting the Earth and in Earth's vicinity, the use of the GNSS constellation, using the pseudorange observations, is very common as was described earlier. For spacecraft operating at lunar distances and beyond, however, the observation data are different. In this case, the primary data used to determine their orbits are two-way radio signals sent from antennas on the ground. The spacecraft is equipped with a transponder that receives, then coherently retransmits the radio signal back to the ground antenna. This radio signal provides two pieces of information: the LOS velocity of the spacecraft relative to the station (Doppler data) and the LOS range. Additional data types which provide complimentary pieces of information to Doppler and range include Delta Differential One-way Range (DDOR) and optical images from a camera. Because the OD process is always a correction to an a priori orbit, either from a previously designed

reference orbit or from a previous orbit solution, there is usually not observability issues, and any one data type by itself can be used to refine the a priori orbit. For all missions, Doppler and range are always used, with DDOR being used more sparingly and optical data primarily used in situations where the ephemerides of the target Solar System body is highly uncertain. Each of these data types will now be described in a little more detail.

The LOS velocity is obtained by exploiting the well-known Doppler effect. The frequency of the signal received by the spacecraft will be Doppler shifted away from the signal sent by the station due to the velocity of the spacecraft toward or away from the station; as the spacecraft coherently turns the signal around and sends it back to the station, it will be Doppler shifted again. The relationship between the change in frequency and LOS velocity is computed as $\dot{v} = \frac{2f}{c}$, where c is the speed of light. In practice, the Doppler shift at the station is obtained by mixing the known frequency sent against the received frequency; this mixing measures the accumulated change in phase of the transmitted signal relative to the received one. Over a specified interval of time, called the count time, the accumulated phase is divided by the count time to get the velocity change. Typically, the hardware performs the accumulation at 10 Hz, but for most applications, the data are compressed to 60 s to both reduce the amount of data and the point-to-point noise in the data. Also, for deep space applications, the majority of the spacecraft operate in the X-band frequency range, although there is still limited use of S-band, and for some higher precision applications, Ka-band. The range is determined by measuring the time it takes for a known signal to be received after it has been sent to the spacecraft. The shift in phase of the return signal relative to the signal transmitted to the spacecraft, modulus an unknown number of cycles of the signal itself, is proportional to the LOS range.

Delta Differential One-way Range (DDOR) is an interferometric data type. For these data, the spacecraft sends a ranging tone that is simultaneously received by two different tracking stations spaced far apart. The delay in the signal received by one of the stations relative to the other provides information about the angle of the spacecraft relative to the baseline between them. The additional Delta in the DDOR measurement comes from not only measuring the delay of the spacecraft signal but also from a quasar located nearby in the sky. A quasar, or quasistellar radio source, is an extremely luminous active galactic nucleus powered by a black hole; quasars emit large amounts of energy in the form of electromagnetic radiation,

including in the radio spectrum. By doubly differencing the spacecraft delay and the quasar delay, the measurement then becomes the angular separation of the spacecraft from the quasar. This double differencing is used because it removes the common sources of error of the measurement arising from atmospheric effects. The information content of DDOR is the angle of the spacecraft in the plane-of-sky, which is complementary to the LOS information content in Doppler and range data.

The third data type commonly used in deep space applications is optical data. This is typically obtained from an onboard camera taking images of Solar System bodies against a star background. The image of the target body can be a point source that is, the angular extent of the body extends less than a camera pixel (like stars), or resolved, where its extent is greater than a pixel (for example, the Moon as seen from Earth). In either case, methods are used to obtain the precise center of the body, called centerfinding. When stars are in the camera field-of-view, centerfinding is done on the stars as well, and if at least two stars are available, the exact inertial pointing direction of the camera can be computed. The center of brightness of the body then provides a measure of the inertial LOS direction of the body relative to the spacecraft, and thus provides another data source that is complementary to Doppler, range, and DDOR. Optical data are most often used when the orbit knowledge of the target body itself is poorly known, such as the outer planets, planetary satellites, asteroids, and comets. From the above, it can be seen that each type of tracking data provides information along certain dimensions, but individually is not enough to compute the complete state of the spacecraft. By complete state, we mean (in Cartesian space) the three components of position and three components of velocity which describe the state of the spacecraft, relative to the Solar System body in question. Furthermore, typically, the filter will also estimate many other parameters; these include dynamic parameters which affect the motion of the spacecraft (e.g., the solar pressure force acting on the spacecraft, delta-v associated with thruster firings, gravitational effects), as well as parameters that affect the data (e.g., range delays, effects of atmospheric media on the tracking data). In most cases, the filter will also include consider parameters; these are parameters which are not adjusted in the filter but contribute to the overall uncertainty of the filtered solution. The “state” in a filter setup thus describes all the parameters estimated and considered in the filter. For an example of the complete filter list in a standard interplanetary OD filter, see [Table 9.2](#).

Table 9.2 Filter list example in a standard orbit determination filter.

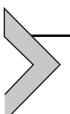
Parameter	Estimated or considered
State (position and velocity)	Estimated
Gravitational parameters (e.g., mass, spherical harmonic coefficients) of central body	Estimated
Solar radiation pressure (overall scale factor or specular and reflectivity values of specific components, like solar panels)	Estimated
Maneuvers	Estimated
Attitude control thruster firings (e.g., momentum wheel desaturation events)	Estimated
Atmospheric radio signal transmission media delay parameters (due to Earth's troposphere and ionosphere)	Considered
Earth orientation parameters	Considered

For ground-based OD, it is often advantageous to use the linearized form of the batch filter (as described above); this method allows straightforward editing of measurements, as well as performing multiple variations on the filter parameters. The variations often can be quite numerous as it is very important to ensure that the filter has converged on the correct solution. Examples of these variations include, but are not limited to, the following:

- The span of time covered by a batch filter estimate, called the “data arc”. Typically, the orbit estimate for the current time in a deep space mission will include multiple filter runs using varying lengths of data arcs; for example, in an interplanetary cruise scenario, long arcs of many months, medium arcs of a few weeks to a month, and short arcs of days to a few weeks will be used to compute solutions and compared for consistency. Also varied will be types of data included, where the estimate is done using each data type described above individually and in different combinations. This is done to ensure that any given data are not corrupted by some unknown bias, which might lead to a bad state estimate. Another variation is the choice of data weighting, where each data type is assigned a different absolute weight or adjusted relative to each other.
- A priori sigma on the state parameters. For example, tight values might be used if a specific parameter is well known, but a variation where the a priori sigma is loosened up to let the filter solve the parameters to the best fit value which may be outside the a priori uncertainty if the initial guess was incorrect. Given the above variations, the OD solution is evaluated carefully to decide on the set of parameters and variations which give the

best filter solution. The solution evaluation is often an art, and it is based on the following criteria:

- The residuals of the tracking data used in the solution should display no significant signatures or trends. The postfit tracking data residuals should display a random scatter about a near zero mean.
- All residual outliers must be removed from the data set used in the solution.
- The data weights should not be less than the observed scatter in the data.
- The estimated parameters should be a sufficient set to accurately model the trajectory and media.
- All the data used in the solution should be sensitive at the appropriate levels to the dynamic and media model errors.
- The corrections to the estimated parameters should make physical sense (e.g., gravitational parameters should not solve out to negative values).
- The a priori uncertainties should be of the appropriate magnitude, in other words, the ratio of the corrections to the parameters to the a priori sigma should not be greater than about 3.
- The scatter of solutions with the variations should be statistically consistent with each other. If not, every attempt should be made to explain the discrepancy, such as examining for unmodeled biases in the data.



Absolute attitude navigation

Attitude navigation, also known as attitude determination, is the task of determining the current attitude state of the satellite for controlling and maneuvering it where necessary. This requires the use of attitude sensors and algorithms that are often custom made for a satellite, as it has strong dependencies from most of the other subsystems. First, let us clarify that all the sensors used to determine the attitude states are based on direction measurements, even though sometimes angles are used, but the underlying concept is the same. Even with off the shelf star sensors, which are capable to independently estimate the full attitude of the spacecraft, we are sure that they work by determining the directions of many stars before computing the result. What they do is comparing the stars they see in the sensor reference frame to a database containing their reference directions in inertial frame. The same process holds for other sensors like magnetometers or sun sensors presented in the [Chapter 6 - Sensors](#): they measure a direction in the sensor frame, which is then rotated by a known mounting angle in the body frame, to be eventually compared with the reference direction in inertial frame.

The latter is often orbit-dependent; hence, a proper orbital determination is crucial to have a reliable attitude reference. Another important point is that at least two different direction measurements are necessary to estimate the complete attitude state. If we fix one direction, we have infinite rotation around that direction that the satellite can take; thus, we need to fix at least another direction that has a significant projection perpendicular to the first direction. This becomes visually clear if we think to a dice, which is also visually represented in the [Chapter 5](#) - Attitude Dynamics. If we “measure” the face 1 to be on top of the dice (i.e., we determine that the face with figure 1 has its normal direction pointing upward), we have four possible orientations that are compatible with this measured condition. We can completely fix the full three-dimensional attitude state if we measure where the face 3 is, for example. However, if we manage to measure the face 6, we know we will not have a full attitude estimation, as figure 1 and 6 are in two opposite faces, and the two directions would be actually the same with opposite sign.

The problem of computing the attitude state from direction measurements is typically referred to as *static attitude determination*, since it is based on geometrical relations between measurements taken in the same instant of time. Static attitude determination has no connection with the attitude dynamics, and different time instants may have uncorrelated attitude solutions. This is the fundamental attitude determination method, but it is not robust with respect to measurement errors and uncertainties. For these reasons, static attitude determination is typically complemented with some filtering techniques, which are sometimes referred to as *dynamic attitude determination*. In fact, these methods use the knowledge of the motion of the spacecraft to accumulate a history of past measurements.

Triad

The simplest attitude determination process requires two direction measurements in the form of unit vector. Indeed, attitude determination requires finding three independent quantities, such as any minimal parameterization of the attitude matrix. Hence, let us assume to measure, for example, the direction of the sun, $\hat{\mathbf{s}}^b$, and of the magnetic field, $\hat{\mathbf{b}}^b$. Since the two vectors change in time depending on the orbit and the day of the year, it is almost impossible to have them perpendicular; hence, a simple way would be to determine other two auxiliary unit vectors starting from these two measurements. The aim of the algorithm is to find the DCM, $\mathbf{A} = {}^b\mathbf{A}_i$, which

rotates the inertial sun direction $\hat{\mathbf{s}}^i$ and magnetic field direction $\hat{\mathbf{b}}^i$ in the body reference frame:

$$\begin{cases} \hat{\mathbf{s}}^b = A\hat{\mathbf{s}}^i \\ \hat{\mathbf{b}}^b = A\hat{\mathbf{b}}^i \end{cases} \quad (9.1)$$

If we construct two vectors set as follows:

$$\begin{cases} \hat{\mathbf{v}}^b = \frac{\hat{\mathbf{b}}^b \times \hat{\mathbf{s}}^b}{\|\hat{\mathbf{b}}^b \times \hat{\mathbf{s}}^b\|} \\ \hat{\mathbf{v}}^i = \frac{\hat{\mathbf{b}}^i \times \hat{\mathbf{s}}^i}{\|\hat{\mathbf{b}}^i \times \hat{\mathbf{s}}^i\|} \\ \hat{\mathbf{k}}^b = \hat{\mathbf{v}}^b \times \hat{\mathbf{b}}^b \\ \hat{\mathbf{k}}^i = \hat{\mathbf{v}}^i \times \hat{\mathbf{b}}^i \end{cases} \quad (9.2)$$

We now have all the elements to determine the DCM. In fact, we just constructed a new reference frame using $\hat{\mathbf{b}}^b$, $\hat{\mathbf{v}}^b$, and $\hat{\mathbf{k}}^b$ whose relationship with the inertial reference is by construction:

$$[\hat{\mathbf{b}}^b \quad \hat{\mathbf{k}}^b \quad \hat{\mathbf{v}}^b] = \mathbf{A} [\hat{\mathbf{b}}^i \quad \hat{\mathbf{k}}^i \quad \hat{\mathbf{v}}^i] \quad (9.3)$$

The simplest way to solve the problem would be:

$$\mathbf{A} = [\hat{\mathbf{b}}^b \quad \hat{\mathbf{k}}^b \quad \hat{\mathbf{v}}^b] [\hat{\mathbf{b}}^i \quad \hat{\mathbf{k}}^i \quad \hat{\mathbf{v}}^i]^T, \quad (9.4)$$

since by construction $[\hat{\mathbf{b}}^b \quad \hat{\mathbf{k}}^b \quad \hat{\mathbf{v}}^b]$ and $[\hat{\mathbf{b}}^i \quad \hat{\mathbf{k}}^i \quad \hat{\mathbf{v}}^i]$ are DCMs.

This operation, however, does not take into account the orthogonality and normality requirements for the solution. Indeed, the solution \mathbf{A} is obtained by numerical means and on-board computers might have not enough precision for the later usage in the GNC scheme. On the other hand, if we make use of a different attitude parametrizations (i.e., Euler angles, Euler axis, quaternions, etc.), the enforcement of the mathematical constraints is easier, but the solution becomes more complex as shown in the next section.

There is a drawback in using this simple formulation, and it is related to measurement noise. If we have an error (note that real measurements are all affected by errors) in $\hat{\mathbf{s}}^b$ and $\hat{\mathbf{b}}^b$, this would affect both $\hat{\mathbf{k}}^b$ and $\hat{\mathbf{v}}^b$. The only

way to improve the solution is to either add more measures or a filtering process:

$$\begin{cases} \hat{\mathbf{s}}_n^b = f(\hat{\mathbf{s}}_1^b, \hat{\mathbf{s}}_2^b, \dots, \hat{\mathbf{s}}_n^b) \\ \hat{\mathbf{b}}_m^b = f(\hat{\mathbf{b}}_1^b, \hat{\mathbf{b}}_2^b, \dots, \hat{\mathbf{b}}_n^b) \end{cases} \rightarrow \begin{aligned} \hat{\mathbf{v}}_e^b &= \frac{\hat{\mathbf{b}}_m^b \times \hat{\mathbf{s}}_n^b}{\|\hat{\mathbf{b}}_m^b \times \hat{\mathbf{s}}_n^b\|} \neq \hat{\mathbf{v}}^b \\ \hat{\mathbf{k}}_e^b &= \hat{\mathbf{v}}_e^b \times \hat{\mathbf{b}}_m^b \neq \hat{\mathbf{k}}^b \end{aligned} \quad (9.5)$$

Wahba problem

We have seen a solution using only two measurements assumed to be equally precise. This is often not the case, as sensors may have very different precision levels, or more than two information are available at the same time. The generalization of this problem is often called Wahba problem [20], first posed by Grace Wahba in 1965, aiming to find a DCM such that minimizes a certain cost function.

Let us identify n body measured directions $\hat{\mathbf{v}}_j^b$ with j going from 1 to n and their inertial counterparts $\hat{\mathbf{v}}_j^i$, the cost function is the following:

$$J = \frac{1}{2} \sum_{j=1}^n \xi_j \|\hat{\mathbf{v}}_j^b - \mathbf{A}\hat{\mathbf{v}}_j^i\|^2 \quad (9.6)$$

where the weights ξ_j can help to discriminate and differently weight the available measurements.

The TRIAD method enforced the measurements to give orthogonal results, but some easy solutions to the Wahba problem might not give as output a real DCM. For example, we might construct two matrices that have n columns, one for each measure, and use the pseudoinverse (noted with the \dagger) to have a least squares solution:

$$\begin{cases} [\hat{\mathbf{v}}_j^b] = A[\hat{\mathbf{v}}_j^i] \\ A = [\hat{\mathbf{v}}_j^b][\hat{\mathbf{v}}_j^i]^\dagger \end{cases}, \text{ with } j = 1:n \quad (9.7)$$

In this case, we have that the minimum number of independent nonparallel measures is three, to allow the inverted matrix to have rank 3. By incorporating the ξ_j , we would have a weighted least square solution. These solutions do not enforce \mathbf{A} to be orthogonal, and thus an orthogonalization procedure would have to be put in place. In fact, the inversion of a

nonsquare matrix using a pseudoinverse cannot generate a perfect inverse; hence, the outcome will, in general, not be an orthogonal matrix representing the DCM. Well-known orthogonalization techniques are the Gram–Schmidt projection and the Householder reflection, but other options exist. The interested reader can find the algorithms in most linear algebra books, like Ref. [21].

Algorithms for solving the Wahba problem are divided into two groups. Some of them solve for the attitude matrix directly, and others solve for the quaternion representation of the attitude matrix. Quaternion solutions have proven to be much more useful in practice, and they are common standard in modern GNC applications.

SVD method

The first formal solution of the Wahba problem is the Singular Value Decomposition (SVD) method to estimate directly the DCM. We can rewrite the cost function (sometimes referred to as loss function) as follows:

$$J = \sum_{j=1}^n \xi_j - \text{tr}(\mathbf{AB}^T), \quad (9.8)$$

where we have defined the matrix \mathbf{B} as follows:

$$\mathbf{B} = \sum_{j=1}^n \xi_j \hat{\mathbf{v}}_j^b (\hat{\mathbf{v}}_j^i)^T. \quad (9.9)$$

This matrix can be decomposed through the SVD, and its left and right matrices can be coupled to get \mathbf{A} :

$$\begin{cases} \mathbf{B} = \mathbf{S}\mathbf{V}\mathbf{D}^T \\ \mathbf{A} = \mathbf{S} \text{diag}([1 \ 1 \ \det(\mathbf{S})(\det(\mathbf{D}))]) \mathbf{D}^T \end{cases} \quad (9.10)$$

An alternative solution would be:

$$\mathbf{A} = \mathbf{B}(\mathbf{B}^T \mathbf{B})^{-0.5} \quad (9.11)$$

That requires \mathbf{B} to be rank 3, meaning to have at least three measurements. In cases when only two measurements are available, it is possible to exploit the same procedure used in the TRIAD and still find a solution with this method.

Davenport q -method

A nicer way to solve the Wahba problem bypassing the passage from DCM to simpler attitude representations is to use directly one of these attitude representation for the estimation. The simplest and most common way is to use quaternions. The first method of this kind is the one proposed by Davenport [22].

Let us identify n measured directions $\hat{\mathbf{v}}_j^b$ with j going from 1 to n . Our goal is to find $\tilde{\mathbf{q}}$ such that the following expression is true for all n measurements:

$$\hat{\mathbf{v}}_j^b = \mathbf{A}(\tilde{\mathbf{q}}) \hat{\mathbf{v}}_j^i. \quad (9.12)$$

Given any possible $\tilde{\mathbf{q}}$, we have that the reprojection error, e_j , for the j -th measurement is:

$$e_j = 1 - (\hat{\mathbf{v}}_j^b)^T \mathbf{A}(\tilde{\mathbf{q}}) \hat{\mathbf{v}}_j^i \quad (9.13)$$

The total error of the set would be:

$$e = \sum_{j=1}^n e_j = n - \sum_{j=1}^n \left((\hat{\mathbf{v}}_j^b)^T \mathbf{A}(\tilde{\mathbf{q}}) \hat{\mathbf{v}}_j^i \right). \quad (9.14)$$

We can express \mathbf{A} as the multiplication of two 4×4 matrices function of $\tilde{\mathbf{q}} = \{ \mathbf{q}_{1:3}^T \quad q_4 \}^T$, and treat the measured and reference directions as unitary quaternions with null real part:

$$e_j = 1 - \begin{Bmatrix} \hat{\mathbf{v}}_j^b \\ 0 \end{Bmatrix}^T \begin{bmatrix} [\mathbf{q}_{13}]^T + \mathbf{I}_3 q_4 & -\mathbf{q}_{13}^T \\ \mathbf{q}_{13}^T & q_4 \end{bmatrix} \begin{Bmatrix} \hat{\mathbf{v}}_j^i \\ 0 \end{Bmatrix}. \quad (9.15)$$

By virtue of quaternion multiplication properties, we get:

$$e_j = 1 - \begin{Bmatrix} \mathbf{q}_{13} \\ q_4 \end{Bmatrix}^T \begin{bmatrix} [\hat{\mathbf{v}}_j^b] & -\hat{\mathbf{v}}_j^b \\ \hat{\mathbf{v}}_j^{bT} & 0 \end{bmatrix} \begin{bmatrix} [\hat{\mathbf{v}}_j^i] & \hat{\mathbf{v}}_j^i \\ -\hat{\mathbf{v}}_j^{iT} & 0 \end{bmatrix} \begin{Bmatrix} \mathbf{q}_{13} \\ q_4 \end{Bmatrix} \quad (9.16)$$

The total error of the set would be:

$$\begin{aligned} e = n - \left\{ \begin{array}{c} \mathbf{q}_{13} \\ q_4 \end{array} \right\}^T \mathbf{Q} \left\{ \begin{array}{c} \mathbf{q}_{13} \\ q_4 \end{array} \right\} &= n - \tilde{\mathbf{q}}^T \mathbf{T} \mathbf{Q} \tilde{\mathbf{q}} = n \left(1 - \tilde{\mathbf{q}}^T \mathbf{T} \frac{1}{n} \mathbf{Q} \tilde{\mathbf{q}} \right) \\ &= \tilde{\mathbf{q}}^T (\mathbf{I}_{4n} - \mathbf{Q}) \tilde{\mathbf{q}}, \end{aligned} \quad (9.17)$$

with:

$$\mathbf{Q} = \sum_{j=1}^n \mathbf{Q}_j = \sum_{j=1}^n \begin{bmatrix} [\hat{\mathbf{v}}_j^i] [\hat{\mathbf{v}}_j^i]^T + \hat{\mathbf{v}}_j^b \otimes \hat{\mathbf{v}}_j^i & \hat{\mathbf{v}}_j^b \times \hat{\mathbf{v}}_j^i \\ (\hat{\mathbf{v}}_j^i \times \hat{\mathbf{v}}_j^b)^T & (\hat{\mathbf{v}}_j^b)^T \hat{\mathbf{v}}_j^i \end{bmatrix}. \quad (9.18)$$

From this, it is clear that finding the minimum of e means finding the maximum of $\tilde{\mathbf{q}}^T \mathbf{Q} \tilde{\mathbf{q}}$; hence, find the unitary $\tilde{\mathbf{q}}$ that maximize this expression.

If we find an eigenpair $(\lambda_k, \mathbf{q}_{\lambda_k})$ of \mathbf{Q} such that $\mathbf{Q} \mathbf{q}_{\lambda_k} = \lambda_k \mathbf{q}_{\lambda_k}$, and we take our estimation $\tilde{\mathbf{q}} = \mathbf{q}_{\lambda_k}$, we have that the error would be:

$$e = n - \mathbf{q}_{\lambda_k}^T \mathbf{Q} \mathbf{q}_{\lambda_k} = n - \lambda_k. \quad (9.19)$$

From this result, we reckon that the minimum error corresponds to a solution that is equal to the eigenvector of the maximum eigenvalue of \mathbf{Q} , λ_{max} : $\tilde{\mathbf{q}} = \mathbf{q}_{\lambda_{max}}$.

This approach can also take into account a weighting factor ξ_j for each pair of measures and references in a simple way:

$$e = \sum_{j=1}^n \xi_j e_j \Bigg/ \sum_{j=1}^n \xi_j \rightarrow \mathbf{Q} = \sum_{j=1}^n \xi_j \mathbf{Q}_j \quad (9.20)$$

We can also notice how this solution can become computationally not efficient as \mathbf{Q} grows larger with the number of measures or weights. The most computationally friendly formulation would use instead $\bar{\mathbf{Q}} = \frac{1}{n} \mathbf{Q}$ with also $\sum \xi_j = 1$, so that the elements of \mathbf{Q} lies approximately in the range $[-1, 1]$.

It should be noted that it is possible to use this algorithm even in the two measurements case, even though it might be more computationally intense than the quality of the result it can provide.

Historically speaking, matrix \mathbf{Q} has been called by Shuster, the Davenport's matrix as the algorithm was first introduced by Paul Davenport. Shuster himself then developed the following method to make the algorithm computationally faster. In the Shuster's method, the matrix \mathbf{Q} is computed in a slightly different procedure, which is here reported for clarity as it can be computationally more efficient:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{C} - \mathbf{I}_3 \operatorname{tr}(\mathbf{B}) & \mathbf{z} \\ \mathbf{z}^T & \operatorname{tr}(\mathbf{B}) \end{bmatrix}, \quad (9.21)$$

with \mathbf{B} the same matrix defined before and:

$$\left\{ \begin{array}{l} \mathbf{z} = \sum_{j=1}^n \xi_j (\hat{\mathbf{v}}_j^b \times \hat{\mathbf{v}}_j^i) \\ \mathbf{B} = \sum_{j=1}^n \xi_j \hat{\mathbf{v}}_j^b (\hat{\mathbf{v}}_j^i)^T \\ \mathbf{C} = \mathbf{B} + \mathbf{B}^T \end{array} \right. \quad (9.22)$$

The final result is the same (i.e., $\tilde{\mathbf{q}} = \mathbf{q}_{\lambda_{max}}$), but this form was used for the development of the QUaternion EStimation (QUEST) algorithm.

QUEST method

The Davenport method to solve Wahba problem is robust and well suited for many applications, although it might be relatively expensive in terms of computational effort. A faster solution is the QUEST method developed by Shuster [23]. QUEST algorithm has become the most widely used algorithm for solving Wahba problem in modern GNC applications. QUEST algorithm is computationally efficient since it allows to avoid the iterative operations on 4×4 matrices required by Davenport's q-method. Indeed, it is based on iterative scalar formulas followed by straightforward matrix multiplications. QUEST solution is based in solving the characteristic equation of \mathbf{Q} , which is theoretically not the best way to solve eigenvalue problems. Therefore, QUEST is in principle less robust than q-method. However, it has proven to be extremely robust in practice.

Using the eigenvector and eigenvalue relations, we have:

$$\begin{cases} (\mathbf{C} - \mathbf{I}_3 \text{tr}(\mathbf{B}))\mathbf{q}_{1:3} + \mathbf{z}q_4 = \lambda_{max}\mathbf{q}_{1:3} \\ \mathbf{z}^T \mathbf{q}_{1:3} + \text{tr}(\mathbf{B})q_4 = \lambda_{max}q_4 \end{cases} \quad (9.23)$$

Due to the construction of matrix \mathbf{Q} , we know that the largest eigenvalue can be approximated as $\lambda_{max} \approx \sum_{j=1}^n \xi_j = \lambda_0$. Hence, Shuster proposed to solve numerically with a Newton–Rapson method the characteristic equation to find the exact value of λ_{max} with a starting solution equal to λ_0 .

If we substitute the first equation of (9.23) into the second equation, we get:

$$(\text{tr}(\mathbf{B}) - \mathbf{z}^T(\mathbf{C} - \mathbf{I}_3(\lambda_{max} + \text{tr}(\mathbf{B})))^{-1}\mathbf{z} - \lambda_{max})q_4 = 0, \quad (9.24)$$

which admits solutions for λ_{max} . Note that if $q_4 = 0$, the matrix $((\lambda_{max} + \text{tr}(\mathbf{B}))\mathbf{I}_3 - \mathbf{C})$ is singular, meaning that 180° rotations give problems to this algorithm. To solve this issue, Shuster proposed the method of sequential rotations to avoid this singularity. Further details on the QUEST algorithm and on the method of sequential rotations may be found in Ref. [24].

The strength of the QUEST algorithm goes in the quick convergence of an iterative Newton–Raphson scheme to solve Eq. (9.24) for λ_{max} using λ_0 as first guess. Few iterations are necessary, and sometimes a single iteration is sufficient. Moreover, the approximation $\lambda_{max} = \lambda_0$ is adequate in many cases. The characteristic equation has an analytic solution, but this solution is slower, no more accurate, and sometimes less reliable than the Newton–Raphson iteration.

When the value of λ_{max} is known, the optimal estimated quaternion takes the form:

$$\tilde{\mathbf{q}} = \begin{Bmatrix} \text{adj}((\lambda_{max} + \text{tr}(\mathbf{B}))\mathbf{I}_3 - \mathbf{C})\mathbf{z} \\ \det((\lambda_{max} + \text{tr}(\mathbf{B}))\mathbf{I}_3 - \mathbf{C}) \end{Bmatrix},$$

which has to be normalized and enforced to be continuous between two consecutive solutions. Indeed, the quaternion solution may abruptly change sign, and this shall be avoided by forcing a sign change to preserve numerical continuity with the previous instant of time. However, it shall be reminded that two quaternions with opposite signs refer to the same physical attitude state, as discussed in the Chapter 5 – Attitude Dynamics.

Estimation of angular velocity

Many attitude control schemes require the estimation of the angular velocity of the satellite, in addition to the determination of the attitude state. This can be attained in different ways depending on the available sensors. If the satellite is equipped with a gyroscope, there is possibility to have a direct measurement; however, gyroscopes are often subject to effects that degrade their performances. The most common problem is a low frequency, almost static, bias error (i.e., bias instability) that can be disastrous for the attitude control. Moreover, there might be the case in which no direct angular rate measurement is available. In all these cases, the estimation of spacecraft's true angular velocity shall be implemented in the attitude navigation functions of a modern GNC subsystem.

Low-cost microelectromechanical system gyroscopes exhibit low-frequency bias instability dependent in a nonlinear, often hysteretic, fashion with temperature and other environmental effects. Also, vibration-induced errors can be accounted for; however, these might be less important during nominal operation once in orbit. In general, if we look at the full spectrum of gyroscope measurements, which has been discussed in the [Chapter 6 - Sensors](#), we notice disturbance errors and noises at different frequencies downgrading the GNC system's performance, as discussed in the [Chapter 10 - Control](#). The low-frequency errors can be estimated if the gyroscope's measurements are inserted in a filter that has estimation capabilities in that frequency range. One possibility is to use one of the attitude estimation methods presented before and confront the variation in time of the estimated attitude with the raw measurement from the gyroscope. In this way, a sequential filter can estimate the gyroscope's bias. Another possibility is to insert the angular rate measurements of the gyroscope in an estimation filter based on the dynamical model of the satellite. In this case, the filter uses the Euler equations and the attitude kinematics to estimate the gyroscope errors. This approach requires a good knowledge of the system, of the external disturbances, and of the commanded actuation. The first possibility, exploiting a filter based on the sensor model, requires less input from the rest of the system and can be quite independent. As one can imagine, the modeling of the entire dynamical system of a satellite, would need information on the inertia of the system, as well as on the control torques that are currently used, which are function of the estimation itself. Instead, if we use a filter that is based on the modeling errors of the sensors (e.g., bias instability, random walks, etc.), this information is no longer needed.

If a gyroscope is not present on the satellite, it is still possible to estimate the angular velocity using other sensor measurements. Indeed, once the attitude is estimated, it is possible to use differentiation to estimate the angular velocity using past and current measurements. This method suffers from high-frequency noise by nature, because of the differentiation operations, and from low-frequency estimation errors, if the original measurement has low update time. A common practice is to use a high-pass filter that in the low-frequency range behaves as a differentiator, plus a low-pass filter to eliminate the high-frequency noise. However, the tuning of such passband filter is not trivial, and it requires a lot of effort along the GNC design.

In Fig. 9.12, a simple low-pass filter is presented. If we take the transfer function between output and input, we have:

$$\frac{y}{u} = \frac{K}{K + s} \quad (9.25)$$

which is a low-pass filter of the first order with cut-off frequency of K , expressed in rad/s (i.e., a cut-off frequency of 100 Hz results in $K = 2\pi \times 100 \text{ rad/s}$). This means that the amplitude of the signal u is preserved up to the cut-off frequency, and then it is attenuated afterward with a 20 dB per decade slope. However, we can also take as output \dot{y} , which is the derivative of y , hence if we multiply by s , we get:

$$\frac{\dot{y}}{u} = \frac{K}{K + s} \quad (9.26)$$

Hence, in the low-frequency portion (i.e., below the cutoff frequency), the signal \dot{y} can be considered equal to \dot{u} . In the high-frequency range, this no longer holds true, and, in practice, there will be noise in the high frequency, like for every differentiation application. From what has been shown, it is straightforward to combine a high pass filter with a lowpass filter to reduce the noise of the differentiator getting something like:

$$\frac{z}{u} = \frac{K_{lp}}{K_{lp} + s} \frac{s}{K_{hp} + s} = \frac{K_{lp}s}{K_{lp}K_{hp} + (K_{hp} + K_{lp})s + s^2} \quad (9.27)$$

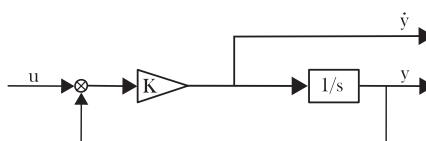


Figure 9.12 First-order low-pass and high-pass filter.

which is a second-order pass-band filter that requires a proper tuning in order to obtain the desired result.

Kalman filtering

We have seen that estimating attitude states and angular velocity is possible, and many navigation solutions and possibilities are available. One common theme is to filter and propagate the measurements and the estimated states to achieve the required update rate, which is linked to the system frequency and error rejection in the complete control loop. The most common technique used in these cases is the KF, previously introduced for absolute orbit navigation. Moreover, Kalman filtering allows improving the stability of the static attitude solution and it allows to estimate unknown quantities and errors (i.e., bias, misalignment, etc.) to improve the overall attitude navigation performance. In general, these techniques are part of the so-called dynamic attitude determination.

If we take the time propagation of the attitude kinematics, as seen in the [Chapter 5 - Attitude Dynamics](#), we can derive a KF around that model to improve the attitude estimation. However, we have seen that such time updates depend on the kinematic parametrizations we use for attitude representation, and in any case, these updates will be nonlinear. This imposes the necessity to use the nonlinear KFs such as the EKF or the UKF [25]. If the computing capacity of the hardware allows for it, also PFs can be used.

In attitude estimation, a further distinction can be made based on the underlining functioning of the EKF. The straightforward implementation of an EKF would see the attitude error defined as a difference between the true and estimated kinematic objects. The simplest solution would be to use Euler angles, but their variation in time is susceptible of singularity, thus they are not recommended for this application. In fact, in modern GNC applications, quaternions are commonly used; however, there are some issues to be addressed. Formulating the attitude error with quaternions as a difference is an operation that does not belong to the unitary quaternion space (i.e., addition is not internal in the unitary quaternion group, as the difference between two unitary quaternions is not in general a unitary quaternion). Hence, it is often preferred to formulate the EKF considering the error to be a quaternion multiplication. This distinction leads to the two main approach: additive EKF (AEKF) or multiplicative EKF (MEKF) [26]. The AEKF neglects the quaternion normalization condition and treats the four components of the quaternion as independent parameters. The

MEKF represents the true attitude as the quaternion product of an error quaternion with the best estimated quaternion: $\mathbf{q} = \delta\mathbf{q} \times \tilde{\mathbf{q}}$, and all of them are normalized unit quaternions. Globally, the MEKF requires less computation resources because it has a lower dimensions covariance matrix, and it also satisfies the mathematical unitary constraint of quaternion rotations. Hence, the MEKF is by far the most used approach for Kalman filtering in modern attitude determination applications.

Another issue is the propagation and inversion of a quaternion covariance. Namely, the four components of a quaternion are not independent, and, thus, the covariance of a quaternion is rank 3 instead of 4. This poses some issues as in the EKF, the covariance of the state is propagated and used in computations. Keeping its rank consistent is not trivial. To solve this issue, one might prefer the use of an UKF where the covariance is computed directly from a subset of propagated states based upon the previous step covariance. During filtering operations, the covariance is computed on perfectly unitary quaternions, since every integration or propagation step has to be normalized anyway, and it can retain all its information without causing divergence. However, UKF is commonly more expensive in terms of computational resources.

A common Kalman filtering approach would be to filter only part of the quaternion component, namely the vector part $\mathbf{q}_{1:3}$, inherently assuming a small error approach. This, of course, cannot be applied directly to the attitude quaternion, as this can assume all possible values. Hence, the filter must be set in a relative error formulation. This means that the KF has to be modeled on the error of estimate. Let us consider the case for MEKF, where we use a gyroscope measure to generate a forward integrator estimate for the attitude quaternion, $\tilde{\mathbf{q}} = \left\{ \tilde{\mathbf{q}}_{1:3}^T \tilde{q}_4 \right\}^T$. Such integrator uses the gyroscope measure, $\boldsymbol{\omega}_g$, and the estimated bias, $\tilde{\mathbf{b}}_g$. In the latter term, we can include offsets and random walks, as well as any low-frequency error source we can think of. The bias estimation can be further expanded to consider dependency on acceleration or temperature, but here these will be omitted for the sake of simplicity. The kinematics of the true attitude parameters is expressed as:

$$\begin{cases} \boldsymbol{\omega} = \boldsymbol{\omega}_g - \tilde{\mathbf{b}}_g \\ 2\dot{\mathbf{q}}_{1:3} = \boldsymbol{\omega} q_4 - \boldsymbol{\omega} \times \mathbf{q}_{1:3} \\ 2\dot{q}_4 = -\boldsymbol{\omega} \cdot \mathbf{q}_{1:3} \end{cases}$$

In this case, the forward integrator will always diverge as it does not take directly into account any attitude measure. The concept at the very core of the MEKF is to define a quaternion error, $\delta\mathbf{q}$, through quaternion multiplication between the real value \mathbf{q} and the estimated value $\tilde{\mathbf{q}}$:

$$\begin{Bmatrix} \delta\mathbf{q}_{1:3} \\ \delta q_4 \end{Bmatrix} = \mathbf{q} \times \tilde{\mathbf{q}}^{-1} = \begin{bmatrix} \mathbf{I}_3 \tilde{q}_4 - [\tilde{\mathbf{q}}_{1:3}]^T & -\tilde{\mathbf{q}}_{1:3} \\ \tilde{\mathbf{q}}_{1:3}^T & \tilde{q}_4 \end{bmatrix} \mathbf{q}.$$

The same reasoning can be applied to the angular velocity error as $\delta\boldsymbol{\omega} = \boldsymbol{\omega} - \tilde{\boldsymbol{\omega}}$. The error variations in time are then given by the following expression:

$$\begin{cases} \delta\dot{\boldsymbol{\omega}} = \dot{\mathbf{b}}_g \\ \delta\dot{\mathbf{q}}_{1:3} = \frac{1}{2} \delta\boldsymbol{\omega} - \tilde{\boldsymbol{\omega}} \times \delta\mathbf{q}_{1:3} \delta\dot{q}_4 = 0 \end{cases}$$

Then, we can construct a KF using the state $\mathbf{x} = \left\{ \delta\mathbf{q}_{1:3}^T \quad \mathbf{b}_g^T \right\}^T$, since we can assume $\delta q_4 \rightarrow 1$ if sufficiently close to the real quaternion \mathbf{q} . In this way, we can use the assumption that the model error covariance \mathbf{Q}_k is a 6×6 matrix that includes the expected bias error covariance. Less trivial is the setting of the covariance related to $\delta\mathbf{q}_{1:3}$, as the model error is theoretically related only to the small angle assumption. The KF would not be complete without the measurements that will increase the understanding of the error and, thus, provide a full-fledged estimation of the attitude. There are two common alternatives depending on the sensors available: one is to exploit the complete attitude estimation in quaternion form of a star tracker; the other is to exploit different direction measurements like Sun, magnetic field, Earth horizon, and so on, to be first processed by a static attitude determination method. Then, the measurement model has to be written in such a form that it can be easily referred to the error component $\delta\mathbf{q}_{1:3}$ (i.e., no

dependency on $\tilde{\mathbf{b}}_g$ is expected). When dealing with vector measurements, it is important to also have the inertial directions available, in order to construct the estimated directions in body frame by using $\mathbf{A}(\tilde{\mathbf{q}})$.

At every iteration step, the KF estimates both $\delta\mathbf{q}_{1:3}$ and $\tilde{\mathbf{b}}_g$. Hence, the output of the propagation can be updated with the first, remembering the assumption $\delta q_4 \rightarrow 1$, and the angular velocity of the gyroscope can be updated using the bias estimate. Important care should be taken in the integrator initialization, as there are two ways to deal with this problem: one is to reset the integrator at every step (i.e., starting from the latest best estimate); the other is to let the integrator run without inserting the new best estimate in the integration phase. It is clear that the two approaches are quite different and can leave to a different result. In the first formulation, $\delta\mathbf{q}_{1:3}$ is set to zero in the prediction step, as the integrator is assumed to be unbiased (i.e., corrected with previous step before the integration); in the second case, this no longer holds true and the whole propagation evolution has to be considered. Moreover, in the latter formulation, there is a higher risk of integration divergence. The reader is invited to deepen the knowledge about Kalman filtering for attitude determination in Refs. [24,26,27].

Complementary filter

It is possible to greatly simplify the structure of the attitude determination filter by using a simple proportional integral (PI) transfer function on the error $\delta\mathbf{q}_{1:3}$ to obtain a complementary filter [28–30]. In general, a KF based on a nonlinear model is time varying and requires nontrivial computations. Thus, if computational effort, simplicity, and tunability is of great concern, using a complementary filter might be the best solution. This filter is simpler to implement, tune, and does not require solving linear systems or store covariance matrices, although it may introduce some delays if not well balanced.

In Fig. 9.13, the simplified scheme for a complementary filter that uses a gyroscope measure ω_g and a static attitude determination quaternion

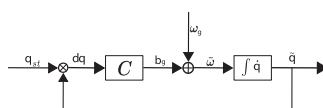


Figure 9.13 Complementary filter scheme.

estimate \mathbf{q}_{st} is presented. Note that the static attitude determination can also be replaced by a star sensor's output. The quaternion error $\delta\mathbf{q}$ is computed using the static estimation and the propagated quaternion estimate $\tilde{\mathbf{q}}$. The latter is computed by integration of the gyroscope measurement plus the bias estimate \mathbf{b}_g , which is the result of a simple proportional and integral control on the vector part of the quaternion error $\delta\mathbf{q}$. A simplified mathematical formulation is proposed here:

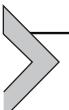
$$\left\{ \begin{array}{l} \tilde{\boldsymbol{\omega}} = \boldsymbol{\omega}_g + \mathbf{b}_g \\ \dot{\tilde{\mathbf{q}}} = \frac{1}{2} \begin{bmatrix} -[\tilde{\boldsymbol{\omega}} \times] & \tilde{\boldsymbol{\omega}} \\ -\tilde{\boldsymbol{\omega}}^T & 0 \end{bmatrix} \tilde{\mathbf{q}} \\ \mathbf{b}_g = C(\delta\mathbf{q}_{1:3}) = K_p \delta\mathbf{q}_{1:3} + K_i \int \delta\mathbf{q}_{1:3} \end{array} \right.$$

This is the resulting simplified model whose dynamics is given by the weights K_p and K_i . These values allow to set a high-pass filter on the gyroscope measures that cuts off their low-frequency content (e.g., bias and random walk) and take instead the low frequency of the derivative of the measurements. This permits to cut off the high-frequency errors in the measurements as well. By delaying $\tilde{\mathbf{q}}$, we can also consider delays in the reference estimated quaternion by other means, which may be interesting in case of star tracker with significant time delay. In fact, in the complementary filter loop, it matters most that the error is computed by comparison of the forecast and the measurement with time coherence. Hence, if we delay the estimation of 1s and we insert a measure with 1s delay, the result will still hold regardless of the delay, although some limitations may apply depending on the sensors used. Another interesting feature of this filter is that it is possible to raise the attitude estimation frequency up to the gyroscope frequency. This means that if a gyroscope has a 10 Hz update frequency, and the static attitude determination has 2 Hz update rate, the output of the filter will be at 10 Hz. Depending on the frequency difference, the final result may vary, and a dedicated analysis on the filter design is always needed.

If we linearize the system in terms of $\delta\mathbf{q}_{1:3}$, we can determine the cut-off frequency and the delay in the measurements as:

$$\omega_{cut-off} \approx \frac{K_p}{K_i + 1}.$$

If we assure $K_i \ll K_p$, the latter is the dominant factor, and it becomes easy to set the cut-off frequency of the filter in such a way that it combines well with the rest of the GNC system.



Relative navigation

The word *navigation* usually refers to absolute spacecraft navigation, i.e., the problem of localizing itself with respect to a known inertial reference frame. In this chapter, the concept of relative navigation is introduced. Relative navigation can be seen as the problem of finding the relative location and attitude of two different space objects' reference frames (i.e., chaser and target). This kind of navigation is of interest to a variety of applications, namely FF, rendezvous and docking (RVD), on-orbit servicing (O-OS) of functional satellites or space station, and active debris removal (ADR). Indeed, in such mission scenarios, the on-board processing unit of one spacecraft must be able to autonomously estimate its relative state with respect to the other, ensuring both high accuracy and update rates. Thus, it shall be able to satisfy control requirements and minimize collision risks. Depending on the category of target, different application scenario can be identified [31]. [Table 9.3](#) reports different possible mission scenarios with the associated chaser and target hardware for each case.

In the actively cooperative target case, both chaser and target have the knowledge of their own position and they exchange information by means of a communication link. In this case, the navigation performance is usually very high, and this solution can be applied to FF and O-OS scenarios in which high accuracy is required. In some cases, the target may also be cooperating in a passive way, through artificial markers on the spacecraft body that can be detected and tracked by the chaser spacecraft. Also in this case, very accurate relative navigation can be performed. When dealing with uncooperative targets, the navigation performance inevitably degrades because of the lack of information provided by the target spacecraft. For this scenario, passive or active sensors have to be used along with advanced software techniques to derive a relative state estimate. Specific attention is also addressed to the case of uncooperative targets which are particularly difficult to approach. In fact, the lack of any a priori knowledge of the target body and the high uncertainty on its motions make the relative navigation problem particularly difficult to tackle. Consequently, advanced, ad hoc, technological, and algorithmic solutions shall be envisaged. With regards to the technological aspects, electro-optical sensors have been identified as the

Table 9.3 Relative navigation scenarios.

Target category	Chaser hardware	Target hardware	Mission scenario
Actively cooperative	RF/GPS antennas	RF/GPS antennas	FF, O-OS, RVD
Passively cooperative	Relative sensors (e.g., cameras, LIDAR)	Artificial markers	FF, O-OS, RVD
Uncooperative known	Relative sensors	—	ADR, O-OS, RVD
Uncooperative unknown	Relative sensors	—	ADR, small-body approach

best option for relative navigation purposes when close-proximity maneuvers (e.g., RVD) toward uncooperative targets are required [31]. In particular, either active Light Detection And Ranging (LIDAR) systems or passive monocular and stereo cameras can be used. The selection of the navigation sensor must consider the resources available on-board in terms of mass, electrical and processing power, on one side, the mission scenario and the costs to be sustained for design and development of the satellite system, on the other side.

From the navigation filter implementation point of view, the difference between absolute and relative estimation is not substantial. However, some alternatives exist while designing a relative navigation filter. The first possible choice concerns the filter architecture. In fact, especially when dealing with multiple spacecraft (FF), there are two possible strategies: centralized or decentralized filters. A centralized filter estimates the state of all the accessible spacecraft performing propagation and measurements update using only a single filter. In this case, all the measurements are sent from each spacecraft to a single processor. The main drawback of this architecture is that when the number of satellites is high, the state vector size becomes significant, and this implies a critical increase of the computational load. In a decentralized approach, each spacecraft performs its own state estimation. Then, the solutions are sent to a “master” spacecraft that performs an estimation of the entire formation. In this case, the computational load does not depend on the size of the formation.

The choice of the dynamical propagation can also affect the overall relative navigation filter performance. In fact, the relative dynamics of two spacecraft can be described by using relative or absolute formulations (cfr. [Chapter 4 - Orbital Dynamics](#) and [Chapter 5 - Attitude Dynamics](#)). Also in this case, the choice between relying on a relative formulation rather than on an absolute one strictly depends on the kind of problem, sensors adopted, and performance to be achieved. Three main different approaches can be considered:

- *Absolute representation.* Generally speaking, absolute propagators are more reliable and configurable for high-fidelity propagation (possibility to include models for drag, SRP, ephemerides of other gravitational bodies, relativistic effects, etc.). This choice is preferable when high performance is required or when most of the measurements are absolute.
- *Relative representation.* Classical relative dynamics formulations rely on simple equations (even providing closed-form solutions) but with the cost of stringent assumptions. When the underlying assumptions of these models (usually circular orbits and close distance) are satisfied, relative formulations for describing the dynamics are preferred. This is also true when most of the measurements are relative and the requirement on relative state estimation accuracy is not so rigid.
- *Absolute/relative representation.* In this case, the dynamical propagation is done in an absolute sense, but the covariance is propagated using the corresponding relative formulation (see Ref. [32] for a detailed description). Mathematically this formulation is equivalent to the absolute one, but computational considerations can be carried out. In fact, the absolute representation shows lower computational load and higher precision for covariance matrix propagation with respect to an absolute/relative formulation [32]. On the other hand, when mostly relative measurements are used, the absolute/relative version of the measurement updates is preferred.

To give a practical example, let's assume to have a rendezvous scenario in which a certain sensor (e.g., camera, LIDAR) gives measurement of relative position between the chaser spacecraft and the target. In this case, a sequential filter can be used with the following parameters:

- *Centralized filter.* If the target is uncooperative, this is the only option because no information can be exchanged between the two spacecraft. Furthermore, for RDV scenarios, this approach is preferred since, often, only one of the two spacecraft (the target) can actively maneuver. Therefore, all the information is stored and available to take a decision.

- *State vector.* As previously explained, two main approaches exist. In this specific case, the state vector can be expressed as:
 - *Absolute representation:* $\mathbf{x} = [\mathbf{p}_C, \mathbf{v}_C, \mathbf{p}_T, \mathbf{v}_T]^T$ where $\mathbf{p}_C, \mathbf{v}_C$ are the position and velocity of the chaser and $\mathbf{p}_T, \mathbf{v}_T$ of the target expressed in an inertial reference frame.
 - *Relative representation:* $\mathbf{x} = [\mathbf{p}_{C_R}, \mathbf{v}_{C_R}]^T$ where $\mathbf{p}_{C_R}, \mathbf{v}_{C_R}$ are the relative position and velocity of the target with respect to the chaser reference frame.
- *Dynamical model.* The dynamical model for the state propagation can be relative (e.g., Clohessy–Wiltshire equation – see [Chapter 4](#) – Orbital Dynamics) or absolute (e.g., Newton’s law – see [Chapter 4](#) – Orbital Dynamics). The choice of the appropriate model is discussed before, but it is highly dependent on the selection of the state vector representation.



Image processing techniques

Cameras are usually used as sensors for spacecraft navigation, especially for relative scenarios, being lightweight and relatively cheap. This kind of active sensor provides an image as output that has to be processed to be used by the navigation filter. This is valid for spacecraft navigation but also for all the other possible applications involving cameras. Also for this reason, image processing is an active and broad research field. The objective of this book is to give an overview of the most common techniques and algorithms used in the space domain and to categorize them depending on the potential application.

Image representation

Usually, a mathematical model is used to represent and describe an image. For most of spacecraft navigation applications, a scalar function is sufficient to describe a monochromatic image. If we consider our eye or a camera sensor, an image is inherently two dimensional. A simple way of representing an image is through a continuous function $f(x, y)$ corresponding to the brightness at each image point (with x and y are the coordinates of the image plane). A simple example is shown in [Fig. 9.14](#). This 2D intensity image is the result of a perspective projection of the 3D scene.

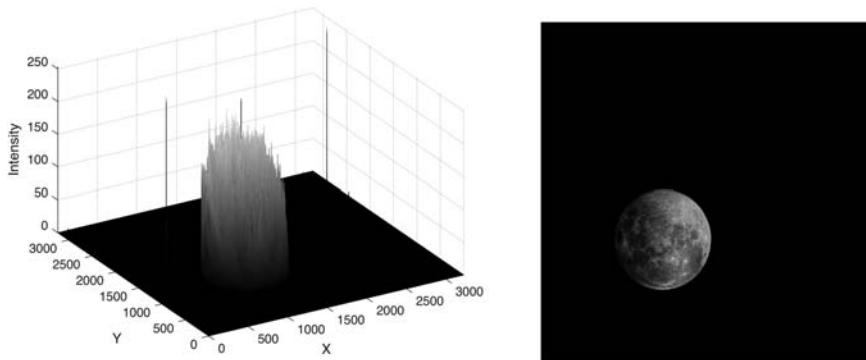


Figure 9.14 Image representation.

Segmentation

The continuous function describing an image can, then, be processed to extract useful image properties. One of the most common image processing techniques is called segmentation. The segmentation process consists in partitioning an image into meaningful regions. This technique is very useful for space applications where a bright object (e.g., a satellite) has to be distinguished from the starry background.

Autonomous segmentation can be a very difficult task for common computer vision applications, but it is less demanding when dealing with simpler space images, where the background is considerably different from the observed target. The most common approaches to segmentations are:

- Local methods – pixel based.
- Global methods – region based.

The pixel-based approach works by detecting the edges within an image and then linking them to create a uniform boundary. The region-based approach searches for pixels with common features and groups them into regions of uniformity.

Local methods

Local methods work by detecting changes of intensity in the image. These methods perform checks to detect the presence of a change across each pixel within an image array. Thus, only local information is used, and the general properties of the whole region are not considered. A good local segmentation algorithm has to rely on a precise edge detector, usually based on gradient operators (for more details, see Ref. [33]).

Global methods

Global methods work by grouping together pixels with common features into regions of uniformity. These methods perform well with high contrast images of an uncluttered scene (i.e., space images). However, they can be computationally expensive for more complex scenarios. Furthermore, global methods tend to generally be less sensitive to noise than local approaches [33]. One of the most common techniques for space application is called thresholding. The idea behind this global method is simple: replace each pixel in an image with a black pixel if the pixel intensity value is lower than a given threshold T . The value of T can be a constant global value depending only on the intensity value of pixel, or it can be updated automatically. One of the most common automatic thresholding techniques is called Otsu's method [34], and it determines the threshold automatically by minimizing the intraclass intensity variance. An example of segmentation using local thresholding with Niblack method [35] (center) or the global Otsu's method (right) is shown in Fig. 9.15.

Fig. 9.15 shows how global methods are usually preferable for space applications since the output is much more uniform and robust to local noise.

2D shape representation

Image segmentation offers a useful tool to define homogeneous regions of an image. However, in most of the cases, it is necessary to identify and describe a given 2D shape. In general, a 3D object can be represented in a 2D plane and a 2D analysis can be performed on a single image. This simplifies the general case of a 3D object reconstruction where multiple images are necessary. Several methods exist to represent a 2D shape, and some of

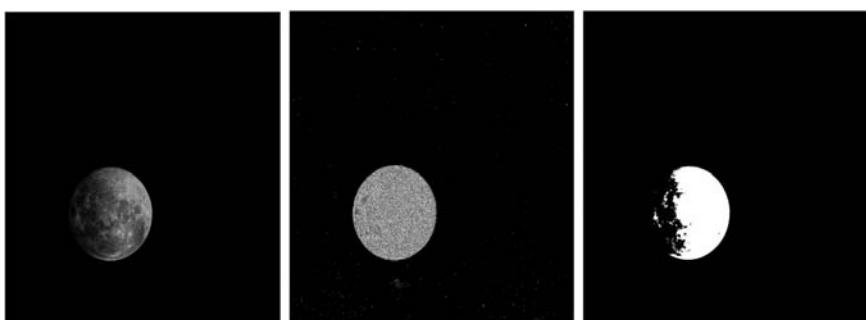


Figure 9.15 Image Segmentation—original image (left), local method (center), global method (right).

them are tailored for specific applications. These approaches can be very useful for space application when, for example, an image of the target should be matched with a given database. In this paragraph, an overview of the main categories of 2D descriptors is presented.

Contour-based shape representation

Mathematical tools have been developed in computer vision to describe contour and boundaries. These descriptors can be of different forms:

Chain codes

In a simple form, a boundary can be represented by a “chain” of connected steps of known direction and length. These steps take the form of unit-size line segments with a given orientation. This definition of chain code was first provided by Freeman [36].

Geometric boundary-based features

The output of chain codes process can be used to derive simple geometrical features. Horizontal, vertical, and diagonal elements can be derived depending on their orientation. This allows to derive the closed-boundary length or perimeter of the 2D shape. In a similar way, the area of the shape can be determined, summing the single vector contributions. Finally, a shape factor parameter can be derived as the ratio between the parameters square and the area. This dimensionless parameter (therefore scale invariant) is useful to measure the elongation of the shape.

Fourier transforms

Another way of describing boundaries is through Fourier transforms. In fact, a complex function can be obtained by assuming to travel along a closed curve with a constant speed. If this speed is chosen such that a complete revolution is performed in time 2π , Fourier transforms can be applied on the obtained period function. The obtained Fourier descriptors can be invariant to translation and rotation in particular cases (see Refs. [37,38]).

Region-based shape representation

For more complex shapes and objects, region-based representation can be used.

Scalar region descriptors

Very simple heuristic descriptors can be employed to describe a given region of an image. These include the area of a given region, its horizontal and vertical projections, its eccentricity or elongatedness, and its direction. Other heuristic descriptors can be also defined or combined, such as rectangularity, being the ration between the region area and the area of a bounding rectangle, or the shape factor described before.

Moments

A very common way to describe regions is by using moment representation. This method interprets an image as a pdf of a 2D random variable. Papoulis [39] proposed to use moments that are statistical quantities to describe the 2D random variable. A generic moment of order $(p+q)$ can be described as:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy$$

where $f(x, y)$ is the pixel brightness and x and y the pixel coordinates. Please notice that this quantity varies depending on scaling, translation, and rotation. Central moments, on the other hand, are translation invariant quantities and are expresses as:

$$m_{pq_c} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - x_c)^p (y - y_c)^q f(x, y) dx dy.$$

Finally, scale-invariant moment can be found by scaling the central moments. A nice discussion on moment invariance is given by Ref. [40].

Applicative case - circular object detection

An applicative case to detect circular objects is discussed in this section.

Centroid detection

Centroid detection techniques are usually used when approaching a far target such as an asteroid, a satellite, or even a planet. In fact, from the position of the centroid in the image, line-of-side measurement can be recovered and used by a navigation filter. The centroid position of an object can be recovered by exploiting chain code techniques [41] or, more often, combining zero and first-order moments. With this technique, often called

Center of Mass technique, the x' and y' coordinates of a centroid can be obtained as:

$$x' = \frac{m_{10}}{m_{00}} \quad y' = \frac{m_{01}}{m_{00}}.$$

In a similar way, Weighted Center of Mass technique combines zero and first-order moments, but weighting them depending on the image brightness.

$$x' = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xf(x, y) W dx dy}{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) W dx dy} \quad y' = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} yf(x, y) W dx dy}{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) W dx dy}$$

with W being an intensity-dependent weighting function that can be modeled as $W = f^p$. The exponent p can be chosen as any real positive number greater or equal to one.

These two simple method and more advanced ones (i.e., Spath algorithm and Hough transform) are presented and compared in Ref. [42].

Limb detection and fitting

In most of the cases, when approaching a celestial body, its shape may not be entirely illuminated by the Sun. This results in capturing only a portion of a circle. In this scenario, before applying centroid detection algorithms, some refinements are necessary. A possible approach is to detect the object limb and to fit it to circles and ellipses and only then compute its centroid. The points belonging to the limb are simply found with an edge-detection algorithm that can be further optimized to take into account atmosphere and local shape of the planet (a very nice description is given in Ref. [43]). The next step is to fit the limb points to a curve: this can be done in several ways, but the two main categories are based on geometrical or algebraic representation of the fitting error. A recent paper by Christian [18] offers a very detailed description of the main curve fitting algorithms and proposes its own horizon navigation algorithm without the need of any conic fit.

3D vision

2D representation is extremely useful and applied for a wide range of scenarios in space. However, especially when dealing with close proximity operations, 3D information of an image became crucial and necessary to correctly perform proximity navigation. 3D vision is usually a very complex and active research field. This paragraph does not pretend to be a complete

guide to 3D vision, but it provides the most important concepts and tools to understand 3D vision and it presents few basic algorithms.

While dealing with 3D vision, two main cases may be discussed:

- *Reconstruction*. Retrieve 3D information of an object by processing a set of images.
- *Recognition*. A priori information of an object is used to perform model-based matching.

Both are treated in this paragraph after an introduction on the main 3D geometry concepts.

Projective geometry

Projective geometry is a set of mathematical tools to describe multiple view geometry. It is used to describe the relations between the 3D points in the scene, their projections on the camera, and the relations among multiple camera projections of a 3D scene. Projective geometry represents the essential mathematical foundation for 3D vision.

Homography

A homography is a projective transformation between two planes or, alternatively, a mapping between two planar projections of an image. In other words, homographies are simple image transformations that describe the relative motion between two images, when the camera (or the observed object) moves. It is the simplest kind of transformation that describes the 2D relationship between two images. Homography can be mathematically described by a 3D transformation in a homogeneous coordinates space and can be expressed as:

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

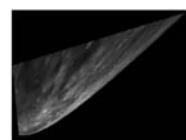
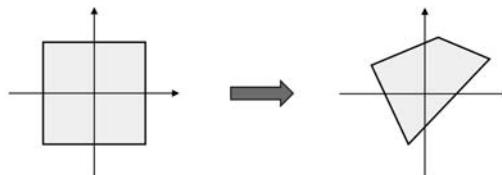
where H allows to transform a 2D point $[x, y]^T$ into an image point $[x', y']^T$.

Several types of homography exist, but the most important ones are summarized in [Table 9.4](#).

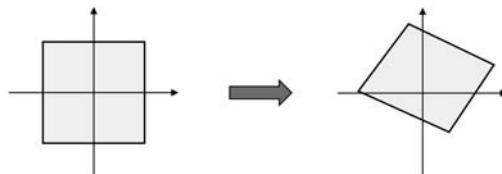
It is important to underline that each homography can be decomposed as $H = H_P H_A H_S$ where $H_P H_A$ represents an affine transformation while H_S represents a metric (also called Euclidean) transformation.

Table 9.4 Projective transformations.**Transformation****2D example**

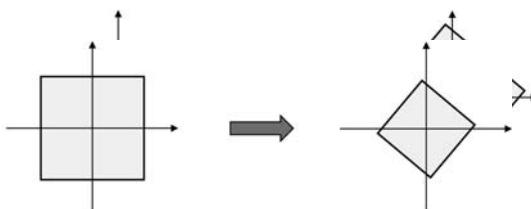
Projective



Affine



Similarity



Metric/Euclidian



Point correspondence-based homography estimation

Given the set of point correspondence between and after the transformation $[x_i, y_i]^T$ and $[x'_i, y'_i]^T$, there exist several methods to estimate the corresponding homography. In an ideal case, where perfect correspondence exists and there is no noise in the measurement, the system is linear and admits a closed form solution. In reality, since the measurements always have a certain level of noise, an optimal estimation problem has to be solved.

Maximum likelihood estimation The estimation problem can be solved by an optimal statistical approach like a maximum likelihood estimation. In this case, the image points are assumed to be random variables with Gaussian distributions, and the objective is to minimize the reprojection error. This kind of optimization process is nonlinear and nonconvex, causing the presence of several local minima. For this reason, a good initial estimate is necessary to guarantee good convergence. The most common algorithm to provide an initial guess and then locally applying maximum likelihood estimation is called Levenberg–Marquardt algorithm [44].

Robust estimation Sometimes, the assumptions of Gaussian noise introduced to perform maximum likelihood estimation do not hold. In these cases, a robust estimator has to be adopted. The standard for robust estimation is the RANSAC algorithm and more details can be found in Ref. [45].

Pinhole camera model

The pinhole camera model approximates the perspective projection of a 3D scene onto a 2D image. In this mathematical formulation, the camera aperture is a point, and lenses, distortion, and blurring are not considered. In other words, all the undistorted ray passes through a point (pinhole), and they are projected onto the focal plane as in Fig. 9.16.

In reality, the pinhole location does not physically exist and, as consequence, the physical camera focal plane and the “apparent” focal plane used by the pinhole model will not be coincident as in Fig. 9.18.

This is due to the fact that a camera is usually composed of an optical assembly, consisting of multiple lenses in front of the camera sensor. For this reason, the arriving rays of light don’t pass through a single point as in the ideal model. To better explain this concept and clarify the pinhole model approximation for a real camera, some optics concepts have to be introduced [18]. The main elements of an optical assembly, considering a recent double Gauss lens, are depicted in Fig. 9.17 and can be summarized as follows:

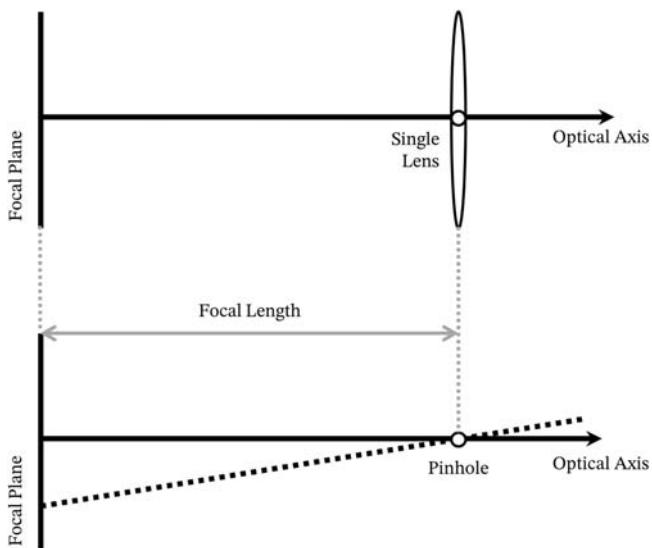


Figure 9.16 Pinhole Camera Model inspired by Ref. [18].

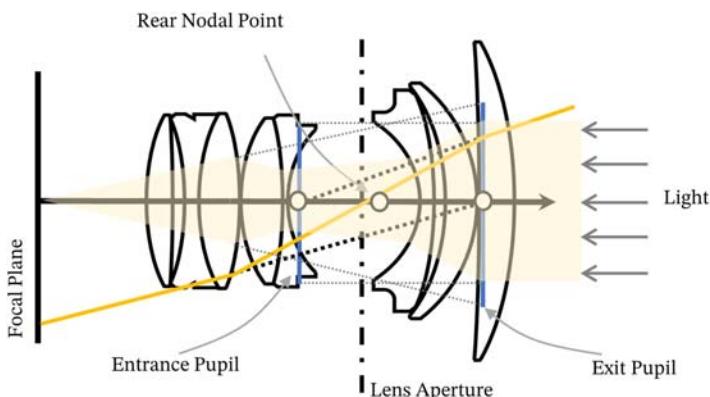


Figure 9.17 Optical assembly schematic.

- *Lens aperture (dashed black line)*. It refers to the opening of the lens' diaphragm through which light passes. This is usually a metallic leaf structure controlling the amount of light which reaches the sensor.
- *Entrance pupil (solid blue line)*. It refers to the optical image of the physical aperture, as seen from the object side of the lens. Its location is found by continuation of the incoming ray of light (solid yellow line) until it

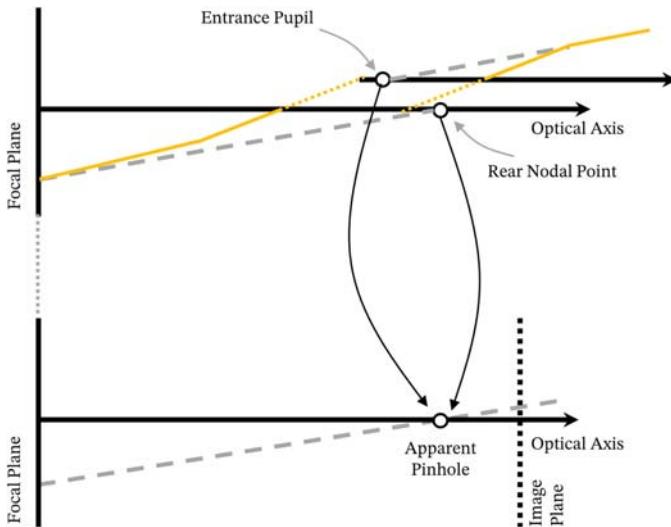


Figure 9.18 Apparent Pinhole and Image Plane, inspired by [18].

intersects the optical axis. Its size, instead, is found by continuation of the incoming light rays (light yellow area) to the entrance pupil location.

- *Exit pupil (solid blue line)*. It refers to the optical image of the physical aperture, as seen from the camera side of the lens. Its location is found by back propagating the ray of light (solid yellow line) until it intersects the optical axis. Its size, instead, is found by back propagating the extremes of the light rays (light yellow area) from the focal plane until the exit pupil location.
- *Rear nodal point*. It refers to the point on the optical axis where the departing ray crosses the optical axis. The departing ray is a ray with the same slope of the incident ray (first segment on the right of the solid yellow line) but starting from the intersection point between the focal plane and the ray of light (see Fig. 9.18).

Fig. 9.17 shows how the apparent pinhole location for an observer outside the camera is the entrance pupil. On the other hand, the center of perspective for an observer inside the camera is at the exit pupil. However, this latter point cannot be used to construct the pinhole camera model since the slope of the rays entering the optical assembly is usually different from the slope of the rays exiting the optical assembly. For this reason, the rear nodal point has been previously introduced, being a point on the optical axis where the slope of the exiting ray is the same of entering one. At this

point, it is clear that to construct the pinhole model for a real camera, the geometries of the inside and of the outside of the camera have to be separated as in Fig. 9.18. That, an apparent pinhole location can be obtained by overlapping the entrance pupil and the rear nodal point. Usually, the inside geometry of the camera can be abstracted, and the camera model is derived in terms of image plane. The image plane is an artificial plane parallel to the focal plane, lying in front of the camera's center of projection. A common convention is to place it at unit dept along the camera optical axis.

Given a 3D point in an arbitrary world coordinate system \mathbf{P}_W , the first step is to transform from arbitrary world coordinate system to camera-centered coordinate system \mathbf{P}_C . This transformation can be written as:

$$\mathbf{P}_C = \mathbf{R}(\mathbf{P}_W - \mathbf{t})$$

where \mathbf{R} and \mathbf{t} describe the rotation and translation, respectively. The composed matrix $[\mathbf{R}|\mathbf{t}]$ is also called *extrinsic camera calibration matrix*. Expressing everything in homogeneous coordinates, we obtain:

$$\mathbf{P}_C = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ 0 & 1 \end{bmatrix} \mathbf{P}_W$$

A second transformation goes from the camera-centered coordinate system \mathbf{P}_C to image plane coordinates $[u_p, v_p]$. Given a 3D point in the camera coordinate system $\mathbf{P}_C = [X_C, Y_C, Z_C]$, the pinhole relationship can be expressed as:

$$u_p = \frac{X_C f}{Z_C}; \quad v_p = \frac{Y_C f}{Z_C}$$

where f is the focal length. The conversion from the image plane coordinates to pixel coordinates is usually performed through an affine transformation. The relationship can be expressed as:

$$u = \frac{f}{\mu_x} u_p + u_0; \quad v = \frac{f}{\mu_y} v_p + v_0$$

where $[u_0, v_0]$ are the coordinates of the focal's plane principal point ($[0, 0]$ is the coordinate system centered in the focal plane) and $[\mu_x, \mu_y]$ the x and y coordinates of the distance between two pixel centers. This relationship can be easily rewritten in homogeneous coordinates as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix}$$

Or simply $x = \mathbf{K}x_p$, with $\alpha_x = f/\mu_x$, $\alpha_y = f/\mu_y$ and γ the skew coefficient between x and y axis. The matrix $\mathbf{K} = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$ is also called *intrinsic camera calibration matrix*.

The pinhole camera projection model in full generality is a combination of the previously described transformations and can be rewritten as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & -\mathbf{R}\mathbf{t} \\ 0 & 1 \end{bmatrix} \mathbf{P}_W = \mathbf{K}[\mathbf{R}|-\mathbf{R}\mathbf{t}]\mathbf{P}_W$$

with $\mathbf{K}[\mathbf{R}|-\mathbf{R}\mathbf{t}] = \mathbf{M}$, being the projection or camera matrix.

Summarizing, this model describes the relationship between a 3D point in an arbitrary world coordinate system \mathbf{P}_W and their two projections in image plane pixel coordinates $[u, v]$. This is done by first converting \mathbf{P}_W into camera-centered coordinate system \mathbf{P}_C through the *extrinsic camera calibration matrix*. Then, the pinhole relationship is used to obtain the 2D projections in image plane coordinates $[u_p, v_p]$ that are finally converted in pixel coordinates using the *intrinsic camera calibration matrix*.

Camera calibration from a known scene

Camera calibration from a known scene is a similar problem to homography estimation, both being homogeneous linear systems subject to noisy measurements. Also in this case, the point correspondences are assumed to be known and the camera calibration matrix can be obtained by applying simple linear estimation [46] or by computing the maximum likelihood estimated.

Multiple views scene reconstruction

Usually, to retrieve 3D information using 2D images and without the previous knowledge of the object being observed, multiple views of the same object are necessary. A simple example is represented in Fig. 9.19.

Triangulation

The simplest case, also called triangulation, is when the camera calibration \mathbf{K} and the image points $[u, v]_i$ are given and the objective is to compute the 3D scene points \mathbf{P}_W . Triangulation resembles once again the homography estimation or the camera calibration matrix estimation problem. Also in this case, the computation of the scene points \mathbf{P}_W closest to all the reprojected image points $[u, v]_i$ can be carried out through a maximum likelihood estimation that minimizes the reprojection error.

Projective reconstruction

If both camera calibration \mathbf{K} and scene points \mathbf{P}_W are unknown, the problem is more complex, and we talk about projective reconstruction. To solve this problem, several images, with their corresponding scene points and camera calibration matrices are needed. Usually, a first estimate of the camera calibration matrix can be obtained by imposing epipolar constraints (in the case of two-views geometry), and then, this first rough estimate of the camera calibration matrix is used to estimate the scene points. This first guess for both calibration matrix and scene points is then used as the initial guess

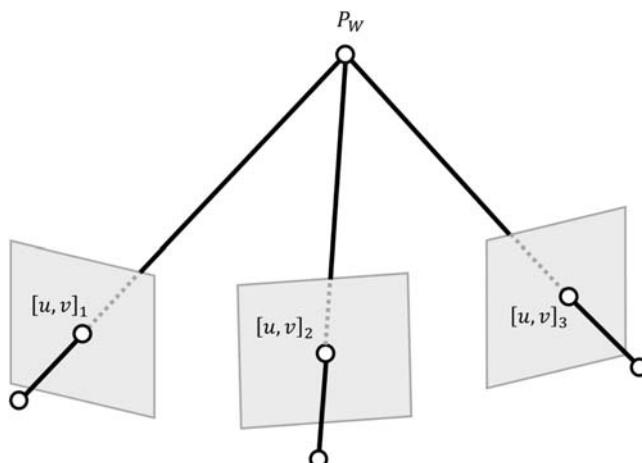


Figure 9.19 Multiple views geometry.

for a nonlinear least square algorithm called bundle adjustment [47]. For more details, please refer to Ref. [48].

Pose estimation

When speaking of vision-based spacecraft relative navigation, the most common applicative scenario is monocular pose estimation. Monocular pose estimation consists in estimating the relative pose of a target spacecraft with respect to a “chaser” spacecraft by relying on 2D images. Contrary to triangulation where the 2D points correspondences are known and the 3D points have to be estimated, in this case, the observer pose should be estimated given a set of 2D points and the corresponding 3D points. The pose estimation method consists in solving a Perspective-n-Points problem. Several methods have been developed during the last decades, and a good overview is given by Pasqualetto [49].

3D vision summary

In this section, several concepts are introduced and a summary, dividing the different methods in subcategories, is here presented to provide the reader with a clear overall picture. Table 9.5 collects all the different methodologies presented in the section, divided depending on their application.

Two-views geometry

Similar to the human visual system, also for space application, it is common to deal with two-views geometry. It is important to underline that this is not only the case of stereovision, when two cameras are used, but also when a single camera allows for 3D reconstruction comparing two different views.

Epipolar geometry

An example of two-views geometry is given in Fig. 9.20.

The two optical centers are connected by the *baseline* that intersects the image planes in the *epipoles*. An epipolar plane can be defined by considering any given 3D scene point \mathbf{P}_W and the two corresponding rays from the optical centers. Let \mathbf{x}_1 and \mathbf{x}_2 be the projections of a 3D point onto the two camera planes, respectively, an algebraic epipolar constraint can be formulated [50] as:

$$\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = 0.$$

Table 9.5 3D Vision summary.

	2D to 2D transformation	3D scene to 2D image	Multiple 2D images to 3D scene		
<i>Method</i>	Homography	Camera calibration	Triangulation	Projective reconstruction	Pose estimation
<i>Known</i>	$[x_i, y_i]^T, [x'_i, y'_i]^T$	$\mathbf{P}_W, [u, v]_i, [\mathbf{R}] - \mathbf{R}\mathbf{t}$	$\mathbf{K}, [u, v]_i$	$[u, v]_i$	$\mathbf{P}_W, [u, v]_i, \mathbf{K}$
<i>Solve-for</i>	H	\mathbf{K}	\mathbf{P}_W	\mathbf{K}, \mathbf{P}_W	$[\mathbf{R}] - \mathbf{R}\mathbf{t}$

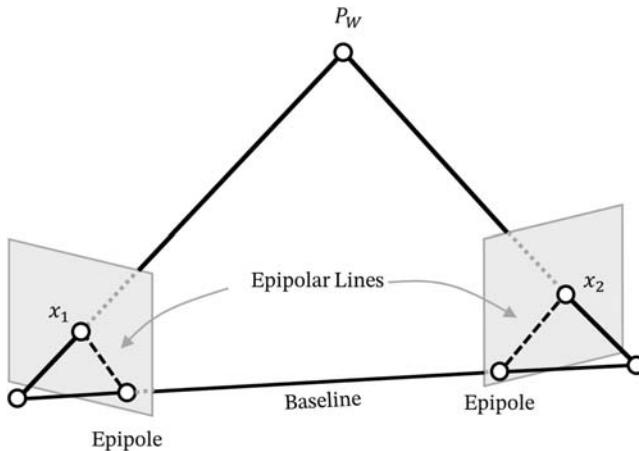


Figure 9.20 Epipolar geometry.

The matrix \mathbf{F} is called *fundamental matrix*, and it is very important since it captures the information between a pair of corresponding points in the two cameras.

At this point, recalling the relation between image plane coordinates and pixel coordinates, for each given camera view, we have:

$$\mathbf{x}_1 = \mathbf{K}_1^{-1} \mathbf{x}_{p1} \text{ and } \mathbf{x}_2 = \mathbf{K}_2^{-1} \mathbf{x}_{p2}.$$

Substituting the previous expression into the epipolar constraint $\mathbf{x}_2 \mathbf{F} \mathbf{x}_1 = 0$, we obtain:

$$\mathbf{x}_{C_2}^T \mathbf{E} \mathbf{x}_{C_1} = 0$$

where $\mathbf{E} = \mathbf{R}^* [\mathbf{t}_x]$ is the *essential matrix*. A nice property of the essential matrix is that it has rank two and its two nonzero singular values are equal. This means that the singular values of the essential matrix are invariant to an orthogonal transformation matrix.

Fundamental and essential matrices have the same purpose, i.e., given a 2D point in one image, they allow to constrain the epipolar line on which the corresponding 2D point in the second image lays. In other words, they both relate corresponding points between a pair of images. The difference between the two is that the fundamental matrix does it in pixel coordinates, while the essential matrix does it using image coordinates. For this reason, the fundamental matrix can be seen as a generalization of the essential matrix when the calibrated camera assumption doesn't hold.

Relative motion of a camera

The relative motion of a camera given two views can be computed with the following steps (please note that this equivalent to the projective reconstruction):

- Compute the fundamental (or essential if the camera is calibrated) matrix from point correspondences.
- Compute the camera matrices \mathbf{M}_1 and \mathbf{M}_2 (please refer to pinhole camera model).
- Given the points correspondence in the two images, compute the 3D point in space that produces the two image points.

Fundamental matrix estimation

The fundamental matrix, given two views, can be estimated knowing at least seven point correspondences. This is because epipolar geometry has seven degrees of freedom. The nine components of the matrix \mathbf{F} are subject to two constraints (a scale factor and that $\det(\mathbf{F}) = 0$), and, therefore, seven free parameters have to be found. If seven correspondences between two images are available, the epipolar problem can be found through the nonlinear seven-point algorithm. On the other hand, if eight points are available, the solution becomes linear.

Eight-point algorithm A simple linear algorithm can be used if there are eight or more point pairs $(\mathbf{x}_{1,i}, \mathbf{x}_{2,i})$ that allow to solve the system $\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = 0$. This problem can be expressed as $\mathbf{x}_2^T \mathbf{F} \mathbf{x}_1 = [\mathbf{x}_2^T \otimes \mathbf{x}_1] \mathbf{f} = 0$ where \otimes is the Kronecker product and $\mathbf{f} = [\mathbf{F}_{11}, \mathbf{F}_{21}, \dots, \mathbf{F}_{23}, \mathbf{F}_{33}]$. If eight correspondences are available, the presented system admits a single solution. Please keep in mind that the obtained \mathbf{F} is nonsingular in general, and therefore, a singular matrix $\widehat{\mathbf{F}}$ can be computed decomposing the fundamental matrix by SVD: $\mathbf{F} = \mathbf{U} \mathbf{D} \mathbf{V}^T$ and setting the smallest value of the diagonal matrix \mathbf{D} to zero. Given the modified $\widehat{\mathbf{D}}$, the new singular fundamental matrix $\widehat{\mathbf{F}}$ can be composed back as $\widehat{\mathbf{F}} = \mathbf{U} \widehat{\mathbf{D}} \mathbf{V}^T$.

Seven-point algorithm If only seven point-correspondences are available, the system admits two solutions \mathbf{f} and \mathbf{f}' satisfying $[\mathbf{u}_2^T \otimes \mathbf{u}_1] \mathbf{f} = [\mathbf{u}_2^T \otimes \mathbf{u}_1] \mathbf{f}' = 0$. The main principle of the seven-point algorithm is to find the points that satisfy the constraint $\det(\mathbf{F}) = 0$. The complete derivation of the algorithm is out of the scope of this book and can be found in Ref. [48].

Camera matrix and 3D point computation

The camera matrices \mathbf{M}_1 and \mathbf{M}_2 can be directly computed from the fundamental matrix. However, the fundamental matrix determines the pair of camera matrices at best up to a 3D projective transformation. This ambiguity can be solved, and different methods have been developed to this purpose. Please refer to Ref. [46] for a detailed derivation. Once the camera matrices are known, the position of the 3D point can be retrieved. However, also in this case, the same ambiguity arises. In other words, the whole scene can be reconstructed up to a 3D homography. Numerical methods have been developed to determine the intersection of the rays corresponding to the two cameras and more detailed can be found in Ref. [46].

Stereo correspondence

So far, we have understood the importance of finding reliable point correspondences between two images to retrieve 3D information. In this paragraph, an overview on how these correspondences are identified is presented. Even though there exist other stereo correspondence methods (e.g., correlation based), the most widely applied method uses features-based correspondence. A feature is a salient point in the image that can be identified across different frames. Examples of feature points are edges, lines, corners, and so on. Several methods exist to extract these feature points, and the most important are detailed in Ref. [51]. Without entering into the algorithmic details, one of the most classical approaches to solve stereo correspondence is the Pollard-Mayhew-Frisby (PMF) method [52]. Given two sets of feature points, the PMF algorithm produces the best correspondences by first enforcing the epipolar constraint between the two images. The score of a match is incremented if other possible matches are found that do not violate a disparity gradient limit. The disparity gradient is an indication of the relative disparity (i.e., difference of the location of an object seen by two different views) between two pairs of points.

Application cases - from subpixel to resolved object

The image processing techniques that are introduced in this chapter can be applied during different phases of spacecraft missions. In this section, a classification depending on the size of the target (in pixel) in the camera frame is provided. Please note that the pixel size on the sensor cannot directly be related to the relative distance if the sensor characteristics are not known.

Subpixel

When the target is very far from the observer or if it is very small, the apparent size of the object on the camera sensor can be equal or less than one pixel. In this case, the only measurement that can be retrieved is the line of sight of the target, described by the 2D coordinates on the camera plane. It is important to underline that, in this case, it is necessary to have a signal-to-noise ratio of the target high enough to obtain a valid measurement and to distinguish the object from background noise. Another aspect to take into account is that, in this scenario, it is not always easy to discriminate between the target and a star in the background. For this reason, it is usually important to collect multiple subsequent frames to distinguish an artificial object from the starry background. In fact, if active pointing is available, the target object will result in a dot of few pixels while the stars will move uniformly in the background. On the contrary, if active pointing is not available, the target object will produce another streak that will most likely have a different orientation with respect to the stars. An artistic and simplified representation of the two scenarios is reported in Fig. 9.21.

Few tenths of pixels

If the target object is sufficiently close or big, its projection on the camera sensor can be large enough to perform some basic image processing techniques. Even if the object projection is few tenths of pixels wide, only LOS information can be retrieved from an image. To this purpose, slightly more accurate algorithms can be used, and basic centroid and limb detection algorithms can be applied.

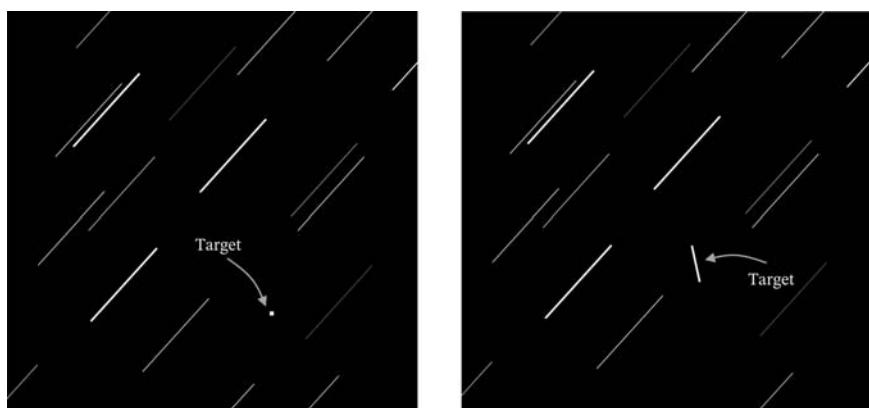


Figure 9.21 Far object detection—active pointing (left), nonactive pointing (right).

Resolved object

When the characteristics of the object are clearly identifiable and the object is resolved in the camera sensor, several techniques can be exploited to extract not only LOS but also pose information. Depending on the type of target, different approaches can be adopted.

Known object

When the object is known, model-based algorithms are usually employed. For simple shapes, the preferred approach is a 2D shape matching technique that can provide accurate pose measurement. When the object is complex, a full 3D model matching technique should be implemented. In this case, the obtained 2D descriptors are matched against the correspond descriptor of the reference 3D model. Once the 2D-3D correspondence is obtained, the pose can be retrieved [53]. Similarly, in the case of relative navigation around a known planet (or moon), the observed features and landmarks observed on the surface can be matched with a known database.

Unknown object

When the object is unknown or information about the approached object cannot be used, different techniques should be employed. One possibility is to rely on stereo vision algorithm [54], able to extract 3D information processing only 2D images acquired with a known baseline. Another possibility is to use Simultaneous Localization and Mapping (SLAM) techniques [55]. As the name suggests, SLAM is a powerful tool to perform localization while building a map of the environment. While this method is extremely common in robotics applications, its use in space application has been limited by the significant computational effort. Moreover, good initialization or loop closure [56] is necessary to achieve satisfactory accuracy and both not easily accessible in space applications. In the case of planetary navigation, if the spacecraft is too close to the surface and, therefore, it cannot distinguish any landmark in the available database, feature-tracking techniques are adopted (see [Chapter 14](#) - Applicative GNC Cases and Examples for more details).

Image processing and spacecraft navigation

In the previous sections, some useful instruments to process images have been introduced. The information acquired by processing an image with the presented methodologies has to be used by the navigation filter to improve the knowledge of the spacecraft state. Also in this case, two main

approaches exist, and we will refer to them as *loosely coupled* or *tightly coupled* architecture.

Loosely coupled. Position and attitude information are computed by an image processing algorithm prior to the navigation filter, and only pose measurement is fed to the navigation. In this way, the filter is not aware of the preprocessing of the image, but it processes only the derived pose information.

Tightly coupled. Centroids, feature points, landmarks, or other image characteristics are directly used as inputs to the navigation filter as measurements. The filter processes this information to provide a state estimate.

Loosely coupled architecture is preferred when the robustness of the image processing algorithm is challenged by the scenario (e.g., fast relative dynamics) leading to highly variable measurements as input to the filter. On the contrary, not many image processing-based pose estimation algorithms provide an indication of the accuracy of the provided solution, leading to a very uncertain value of the measurement covariance and, therefore, low filter robustness if used in combination to loosely coupled architectures. However, tightly coupled methods usually need to process a high number of measurements (e.g., feature points) increasing the overall filter computational complexity. Furthermore, the resulting measurement equation is usually more complex, having to describe the relationship between the state and the image features. The choice of the best navigation filter architecture is highly dependent on the scenario, on the kind of image processing techniques applied, and on the available hardware resources, and a general rule of thumb does not exist. The choice should be based on the general considerations drawn before, considering the specific application.



Navigation budgets

During spacecraft navigation design process, the performance of a navigation filter has to be evaluated. In particular, three elements are usually used to characterize navigation performance:

- Estimation error.
- Robustness to modeled uncertainties and neglected effects.
- Convergence.

These parameters are valid for both batch and sequential navigation filters.

Estimation error and filter robustness

With the term “estimation error”, we define the error between the estimated state $\hat{\mathbf{x}}$ and the reference true state value, \mathbf{x} . At a given time instant k , it can be expressed as:

$$\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k^+.$$

While evaluating the performance of a navigation filter, this quantity is not sufficient for a probabilistic analysis of the filter accuracy. For this reason, Monte Carlo campaigns are usually run to assess navigation performance and filter robustness to uncertain parameters and unmodeled effects. This kind of analyses allows to assess the estimator performance, usually described by the RMSE, which is defined as:

$$RMSE_{MC} = \sqrt{\frac{1}{M} \sum_{i=1}^M \mathbf{e}_{k,i}^T \mathbf{e}_{k,i}}$$

with M being the number of total Monte Carlo simulations. The RMSE is widely adopted as performance parameter because it is the best approximation of the standard deviation of the estimation error. Dealing with probabilistic analysis, the RMSE is the most useful metric to characterize navigation performance. Please, keep in mind that the RMSE exists also for a single filter run where the summation is performed over the simulation time and not over the Monte Carlo runs. In this case, the RMSE takes the

form of $RMSE = \sqrt{\frac{1}{T} \sum_{i=1}^T \mathbf{e}_{k,i}^T \mathbf{e}_{k,i}}$ with T being the simulation time. This

metric can be used, once the filter has converged, to evaluate the steady-state estimation error, but it does not capture the statistical properties and performance of the filter, being computed over a single run.

Similarly, it is possible to define another metric that describes the mean error rather than the standard deviation. This parameter is usually called average Euclidian error or mean absolute error (MAE) and it can be expressed as:

$$MAE = \frac{1}{M} \sum_{i=1}^M \sqrt{\mathbf{e}_{k,i}^T \mathbf{e}_{k,i}}.$$

MAE offers a more immediate and practical concept of error, describing an average absolute error, but it is seldomly used for statistical and probabilistic analysis, during navigation filter characterization.

Please note that, due to the stochastic nature of the filter formulation, in theory, it would not be necessary to perform a Monte Carlo campaign to evaluate the statistical properties of a filter. By definition, in fact, the covariance of the estimation error already embeds the information of the probability distribution of the filter error, i.e., $\mathbf{P}_{i,i} = \text{diag}(\sigma_{i,i}^2)$ with $\sigma_{i,i}$ being the estimation error standard deviation for each state. Strictly speaking, this is true for an ideal linear KF, when the system and observation models and their uncertainties are exactly known. In this case, the estimation covariance converges to a steady-state value that bounds the overall estimation error: $RMSE < \pm 3\sqrt{\mathbf{P}_{i,i}}$. In real cases, with nonlinear models, uncertain dynamical and observation models, the steady-state convergence property of the filter does not hold. However, a common practice with sequential filter is to plot the $\pm 3\sqrt{\mathbf{P}_{i,i}}$ curves (corresponding to the $\pm 3\sigma$ bound) along with the estimation error for a single run of the filter. In fact, if the filter is properly tuned, these curves allow to have a preliminary quantitative idea of the maximum expected estimation error. On the other hand, they can also be used to check the consistency of the filter tuning for a given scenario. In other words, for a given Monte Carlo campaign, it can be assumed that $RMSE_{MC} \approx \sigma_{i,i} = \sqrt{\mathbf{P}_{i,i}}$. A simple example of a generic EKF Monte Carlo campaign of 50 runs and the corresponding $\pm 3\sqrt{\mathbf{P}_{i,i}}$ curves is reported in Fig. 9.22.

It can be observed that the estimation error is bounded by the $\pm 3\sqrt{\mathbf{P}_{i,i}}$ curves for the complete duration of the simulation. Once again, these curves

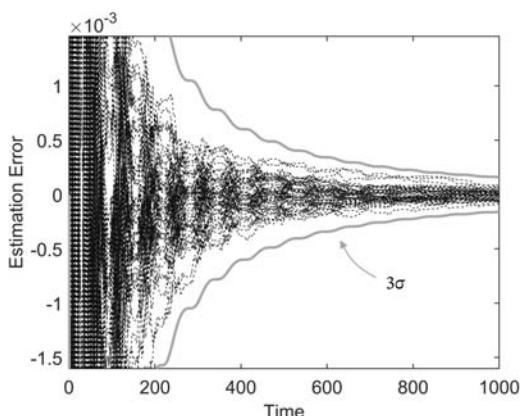


Figure 9.22 50 runs Monte Carlo campaign.

allow to have a preliminary evaluation of the maximum estimation error achievable for a given scenario. Please keep in mind that Monte Carlo analysis is still necessary to validate the filter in a probabilistic sense.

Convergence

The definition of convergence for a sequential or a batch filter is conceptually different. In general, a navigation filter reaches convergence if the estimate error is lower than a threshold. In a batch filter, this evaluation is performed for each algorithm iteration and used as stopping condition. In a sequential filter, the convergence is achieved slowly, processing measurements at each instant of time and, therefore, gradually improving the estimation error. A graphical representation of this difference is depicted in Fig. 9.23.

The convergence threshold can be imposed directly on the estimation error or on the observation residual. During the design phase or postprocessing analysis, it is common to use the estimation error as reference. However, for on-board applications, the estimation error cannot be determined being the “true” spacecraft state not available. For sequential filters, a common practice is to evaluate the speed of convergence in a qualitative way. This qualitative characterization is usually sufficient for most of the spacecraft navigation applications. However, more rigorous ways of quantifying the convergence of a filter exist. These techniques can be used for on-board application. Since the “true” spacecraft state is not available, the observation residual can be used to perform a convergence check. In particular, the observation normalized residual is computed as:

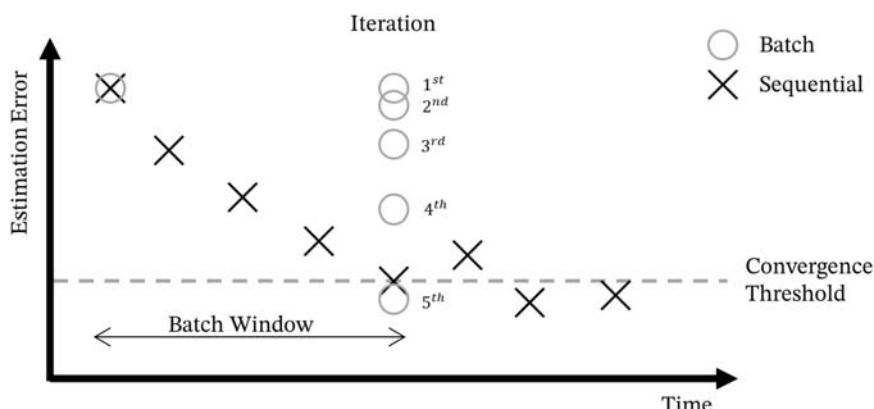


Figure 9.23 Convergence for batch and sequential filter.

$$\mathbf{r}_n = \frac{\mathbf{y}_k - h(\hat{\mathbf{x}}_k^-, 0)}{\sigma_v}$$

with \mathbf{y}_k being the sensor output, h_k the observation model function, and σ_v the measurement noise standard deviation. Convergence is achieved for $-3 \leq \mathbf{r}_n \leq 3$. To give a more practical interpretation of this constraint, it is important to remind that the observation residual quantifies the error between the actual measurement and the measurement predicted by the filter. If the state is estimated correctly (i.e., the estimation error is low), the measurement model will provide a predicted measurement very close to the current value, and, therefore, the observation residual will be small. On the contrary, if the estimation error is high, this will directly result in a large difference between the measurement and its predicted value and, therefore, in a large value of the observation normalized residual. The exact same principle can be used on-board to detect filter divergence and consequently reset the estimation process. Please, keep in mind that this criterion fails if some of the filter states are not observable. In this case, the direct measurement of a given state is not available and, therefore, the state can converge to a value that is far from its true value but still minimizing the measurement residual. Therefore, the residual will eventually be inside the convergence threshold, but the value of estimation error may be very high.

Effect on the GNC chain

During real spacecraft operations, the “true” state of the spacecraft is unknown. The navigation is the functional block of the GNC chain that estimates this value and provides this information to the guidance, control, and mode manager. Let’s recall the high-level working principle of a GNC with the help of Fig. 9.24.

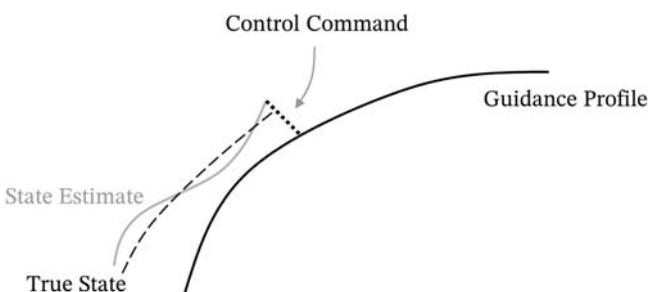


Figure 9.24 GNC chain.

Given a guidance profile and the current spacecraft best estimate provided by the navigation, the control tries to bring the spacecraft as close as possible to the guidance reference trajectory. If we imagine an ideal controller, without any actuator limitation, capable of instantaneously set to zero the error between the estimated state and the reference one, there would be still a residual error due to the navigation performance. With this simple example, it's clear that the navigation poses an intrinsic limit to the accuracy of the whole GNC system. Moreover, high error in the navigation can lead to actuator saturation even in conditions when it would not be necessary, leading to uncontrollability of the spacecraft. Finally, it's easy to imagine the dramatic consequences of undetected filter divergence on the entire mission.

Another potentially fatal outcome of poor navigation performance would be caused by its interaction with the mode manager. In fact, if a given mode is triggered due to wrong state estimation, this could affect the good behavior of the GNC system and jeopardize the mission success.



Navigation implementation best practices

In this chapter, the main practical aspects to take into account while designing and implementing a navigation filter are described.

How to choose the right sequential filter?

There isn't a rule of thumb to select the best sequential filter for a given application, but some aspects may be taken into account while designing spacecraft navigation algorithms:

- *Computational constraints.* Most of the time, the selection of a given sequential estimator is driven by the available computational power. In general, a more accurate estimator implies a higher computational cost, e.g., PFs. However, the available computational power of a spacecraft is usually limited. For this reason, the computational cost of a given solution is, most of the time, the main driver while performing a navigation architecture tradeoff.
- *Dynamical and observation model.* Another critical aspect to consider while designing a navigation filter is to analyze the different available dynamical and observation models, understanding their complexity and accuracy. In particular, if the dynamical or observation models are nonlinear, the choice of the navigation filter will be limited to nonlinear estimators. Moreover, the complexity of a model will drive the selection of a given

estimator. In fact, if the selected dynamical model is very complex, the UKF may not be the best option as it propagates the dynamics for a given number of sigma points and not just once, as done by a standard EKF.

Design, implementation, tuning, and useful checks for sequential filters

In this section, the most important parameters to take into account for a proper design, implementation, and tuning of a sequential estimator are detailed. Furthermore, some useful checks to be performed in order to assess the filter performance are presented.

General design considerations

While designing a navigation filter, some general aspects can be considered:

- *State vector.* The first quantity to define during the implementation of a sequential estimator is the state vector. The state vector collects all the states that have to be estimated, and it has to be carefully selected. A nice feature of sequential estimators is their capability of estimating states that are not directly measurable, e.g., it is possible to estimate velocity by only measuring position. However, it is very important to consider if the elements of the state vector are observable. By definition, an unobservable state is one about which no information may be obtained through the observation model, given the available measurements. If a given state is unobservable, its filter estimate will not converge to a meaningful solution. For this reason, it is important to evaluate the observability of the state vector elements considering the available measurements and observation model.
- *Dynamical and observation model.* In general, it is preferable to select the best models available to describe the dynamical evolution of the state. Still, it is important to take into account their computational load. A simple example is given by the dynamical propagation of a spacecraft orbiting around a given body. The simplest way of describing its motion is by using the Kepler's equation, solution of the restricted two-body problem. Of course, considering other gravitational attractors and other sources of perturbation ([Chapter 3 - The Space Environment](#)) would significantly improve the precision of the state estimate but with a much higher computational load. For this reason, the selection of the appropriate dynamical and observation models is strictly related to the required navigation performance. As generic consideration, recalling the working principle of sequential estimators, if the required navigation performance

is much higher (i.e., small estimation error) than the expected measurement accuracy, the dynamical model precision must be in the same order of magnitude of the required performance. On the contrary, if the expected measurement error is small, it is possible to relax the constraint on the dynamical model.

- *Filter update frequency.* The filter working frequency and the dynamical propagation frequency are usually coincident. In particular, the dynamics update has to guarantee a smooth state transition between one time step and the following one. Having measurements available at higher frequency than the one of the dynamical propagation would imply losing the measurement information (if measurement averaging or buffering is not performed). Depending on the navigation filter application, an appropriate update frequency has to be selected. Fast dynamical environments will, in general, require a higher update frequency.
- *Measurement and their time delay.* Dealing with multiple sensors may lead to have measurements available at different time instants. A sequential filter can usually deal with multiple frequency measurements, but the algorithmic complexity may significantly increase. For real application, the latency of the measurements is very important. In general, measurements need to be processed at the time they are taken, considering their latency. This is usually done by backward propagating the state vector from filter time to measurement time t_m , using the dynamical model. The state at t_m is then used to compute the predicted measurement and measurement covariance at t_m .

Implementation workflow

The main steps to correctly implement a sequential estimator are presented as follows:

- Define the state vector.
- Implement the model for state dynamical propagation. Please note that if the dynamical model is continuous, a state nonlinear discrete propagation has to be implemented. This is usually done by using discrete time integrators such as Runge–Kutta.
- Compute the STM and propagate the state covariance. Please keep in mind that the expression of the STM for linear systems is immediate. However, for nonlinear estimators, it has to be numerically or analytically computed.
- Implement the measurement model for all the available sensors.

- Implement sequential estimator formulae for covariance update and state estimation. If we consider Kalman estimators (KF, EKF, UKF), different ways of performing the measurement update step exist (e.g., Joseph formula, UD factorization). It is important to select the one ensuring filter stability. For more details, please refer to Refs. [2,26].

Implementation efficiency

To improve sequential filter efficiency, some best practices can be followed:

- *Sequential measurements.* If we recall the KF gain computation, $\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$, we can notice that this implies a computational expensive matrix inversion. If the measurements are not correlated, \mathbf{R}_k is diagonal by construction (if not, it can be diagonalized by Cholesky decomposition). In this particular case, the matrix inversion of \mathbf{R}_k can be replaced by a scalar division of the diagonal elements of \mathbf{R}_k .
- *Joseph form.* The use of the Joseph form is a common practice while implementing sequential filters. The main benefit of this approach is that it ensures the positive semidefiniteness of the estimation error covariance matrix at the cost of slightly higher computational effort. Joseph form update can be written as:

$$\begin{aligned}\mathbf{P}_k^+ &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k-1}^- \rightarrow \mathbf{P}_k^+ \\ &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k-1}^- (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)^T + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T\end{aligned}$$

- *Covariance factorization.* This operation usually reduces the arithmetic operations for state and measurement covariance update. The most used covariance matrix factorization method is the UDU technique. The benefits of this method are particularly evident when the state vector is large and, therefore, numerical issues are likely to occur. UDU factorizations are necessary for on-board implementations being numerically stable and computational efficient. The precise derivation of UDU factorization for state and measurement covariance update will not be presented here (the reader can see Ref. [32] for the detailed implementation), but the generic formulas are recalled:

$$\mathbf{P}_k^- = \mathbf{U}_k^- \mathbf{D}_k^- \mathbf{U}_k^{-T} = \mathbf{F}_{k-1} \mathbf{U}_{k-1}^+ \mathbf{D}_{k-1}^+ \mathbf{U}_{k-1}^{+T} \mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$$

and for the measurement update step:

$$\mathbf{P}_k^+ = \mathbf{U}_k^+ \mathbf{D}_k^+ \mathbf{U}_k^{+T} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{U}_{k-1}^- \mathbf{D}_{k-1}^- \mathbf{U}_{k-1}^{-T}$$

with $\mathbf{K}_k = \mathbf{U}_k^- \mathbf{D}_k^- \mathbf{U}_k^{-T} \mathbf{H}_k^T (\mathbf{H}_k \mathbf{U}_k^- \mathbf{D}_k^- \mathbf{U}_k^{-T} \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$.

How to choose the right batch filter?

Even if batch filters techniques are different than the ones employed in sequential estimation, the same high-level principles can be applied in the design and implementation of both types of filters. There are of course small differences that are presented below:

- *Computational constraints.* The main difference is the fact that computational constraints are typically not an issue for batch estimation techniques. As have been mentioned during the navigation chapter in this book, batch filters are usually employed for trajectory reconstruction on-ground. Therefore, the computational constraints inherent to on-board estimation techniques are not present, and, for that reason, batch estimators usually include more complex dynamical and measurement models, as well as higher number of variables and parameters in the state vector.
- *Dynamical and measurement models.* As stated above, since computational constraints are not an issue with batch filtering techniques on-ground, the dynamical and measurement models selected for trajectory reconstruction are typically very high detailed, taking into account a higher number of parameters to model the different source of errors for a given trajectory and, therefore, targeting a higher precision solution. In any case, when selecting the dynamical and measurement modeling, it is important to have a correct balance among all the elements in the filtering process for a given scenario: even if there are no computational constraints, it probably makes no sense to include very detail models of perturbations that are negligible in a concrete situation (e.g., modeling Martian zonals and tesseral in an interplanetary trajectory).
- *Partial derivatives calculation.* Batch filters require the propagation of the measurements to the same epoch in order to proceed with the state update step, as explained before. For this reason, it is important to assure that the partial derivatives to generate the STM are correctly calculated for the complete trajectory arc that is reconstructed. When partial derivatives are calculated numerically, the δ applied to the independent variable needs to be adapted to the dynamical environment: for example, when calculating the derivative with respect to a position component, let's say " \mathbf{x} ," the $\delta\mathbf{x}$ for an interplanetary arc can be in the order of 10 km (have in mind that 10 km are a very small position variation for interplanetary distances that are in the order of hundreds of million km). But, if we are reconstructing descent trajectory to an asteroid,

10 km can be the order of magnitude of the asteroid size. Therefore, in this second situation, the $\delta\mathbf{x}$ can only be in the order of several meters, instead of km. The calculation of partial derivatives is a key factor in batch filtering techniques and a common cause of divergence in the filter if not done properly.

Implementation workflow

In this paragraph, the main steps to correctly implement a batch estimator are presented.

- Define the state vector, including all parameters deemed necessary for the models required in the trajectory reconstruction.
- Implement the model for state dynamical propagation. Trajectory reconstruction models are usually more complex and require higher number of parameters for implementation.
- Implement the measurement model for all the available sensors.
- Compute the STM and propagate measurements to refer them to the same epoch.
- Implement batch estimator formulae for covariance update and state estimation. The selection of the estimator needs to take into account the possible numerical problems that can arise during the filtering process.
- Propagate the state and the covariance until the end of the trajectory arc reconstructed using the numerical propagator and the STM, respectively.

Navigation filters tuning

Tuning usually represents the hardest task while designing a navigation filter. For a person approaching navigation filters for the first time, tuning may seem “*art*” driven by experience. However, if the theoretical principles introduced are clear, some general rules and checks may be identified to properly tuning an estimator:

- *Select initial state value.* If no data are available, it is common practice to randomly select the initial state from a normal distribution with σ extracted from the state covariance matrix \mathbf{P}_0 .
- *Choose the appropriate values for process and measurements covariance matrices.* Conceptually, it may be useful to think about the process and measurement covariance as two parameters to weight how much the filter trusts the dynamical model or the measurements. In general, the selection of the measurement covariance matrix \mathbf{R} is quite straightforward if the sensor sensitivity is known. In fact, \mathbf{R} contains the variance of the available measurements. The selection of the process covariance matrix \mathbf{Q} can

be slightly more complicated. For preflight detailed design studies, the “real world” is simulated, so the expected error of the filter dynamical model with respect to the “real” dynamics can be computed. This is the first iteration value that a filter designer should select as \mathbf{Q} . When the “real-world” simulator is available, it is also possible to run the mission simulation many times, generating an ensemble of parallel results, performing a Monte Carlo analysis. On the other hand, during and after the actual mission, it is not possible to know the “real” state. Therefore, the value of \mathbf{Q} cannot be determined beforehand, but it can be selected iteratively based on the best knowledge of the environment and dynamical model uncertainties.

- *Perform a first check of the dynamical model.* Run in parallel the “real-world” model and the filter without performing any measurement update (i.e., pure propagation), using the same initial condition and check the state difference. This check is useful to verify if the dynamical model inside the filter is implemented correctly, and it also gives an indication of the process error covariance. Once a value for \mathbf{Q} is selected, this step can be repeated, propagating also the state covariance with $\mathbf{P}_k^- = \mathbf{F}_{k-1} \mathbf{P}_{k-1}^+ \mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}$ and check if the error between the “real-world” state and filter propagation state is bounded by the 3σ of \mathbf{P}_k^- . This check provides a first indication on the goodness of the tuning of \mathbf{Q} . This is an important step to avoid the phenomenon of *filter smugness*. As explained before, if the value of \mathbf{Q} is too small, the resulting \mathbf{P} may be too optimistic. In this condition, the filter starts rejecting measurements while trusting only its dynamical model. This may result in stability problems especially for real-world applications.
- *Perform a check on the observation model.* Run the filter update step only, using the “real” state to compute the measurement. Check the error between the real and predicted measurements (i.e., innovation) and see if it is bounded by the selected \mathbf{R} matrix.
- *Perform a final check on the estimation error.* It is common practice to plot the state error (with respect to the “real” state) along with the 3σ of \mathbf{P}_k^+ . In general, a good tuning guarantees that the estimation error is always bounded by the $\pm 3\sigma$ curves.
- *Check residuals.* The difference between “real-world” measurements and “estimated world” measurements are the residuals. If trajectory reconstruction has been performed correctly, residuals should present a flat behavior with average values in the order of the measurement noise and

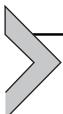
centered around zero, which would imply that the measurement has been modeled correctly and estimated state is close to “real-world” state. If residuals present drifts, or are centered around a different value than zero, this implies that there is a nonmodeled effect in the filtering process, such a bias or an error source that is not taken into account.

References

- [1] V. Pesce, A.A. Agha-mohammadi, M. Lavagna, Autonomous navigation & mapping of small bodies, in: 2018 IEEE Aerospace Conference, IEEE, March 2018, pp. 1–10.
- [2] R.E. Kalman, A new approach to linear filtering and prediction problems, *Journal of Basic Engineering* 82 (1) (1960) 35–45, <https://doi.org/10.1115/1.3662552>.
- [3] R.E. Kalman, R.S. Bucy, New results in linear filtering and prediction theory, *Journal of Basic Engineering* 83 (1) (1961) 95–108, <https://doi.org/10.1115/1.3658902>.
- [4] T. Pappas, A. Laub, N. Sandell, On the numerical solution of the discrete-time algebraic Riccati equation, *IEEE Transactions on Automatic Control* 25 (4) (1980) 631–641.
- [5] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*, John Wiley & Sons, 2006, <https://doi.org/10.5860/choice.44-3334>.
- [6] J. Bellantoni, K. Dodge, A square root formulation of the Kalman-Schmidt filter, *AIAA Journal* 5 (7) (1967) 1309–1314, <https://doi.org/10.2514/6.1967-90>.
- [7] E.A. Wan, R. Van Der Merwe, The unscented Kalman filter for nonlinear estimation, in: *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, IEEE, 2000, pp. 153–158.
- [8] N.J. Gordon, D.J. Salmond, A.F. Smith, Novel approach to nonlinear/non- Gaussian Bayesian state estimation, in: *IEE Proceedings F (Radar and Signal Processing)*, vol 140, IET, 1993, pp. 107–113, <https://doi.org/10.1049/ip-f-2.1993.0015>.
- [9] F. Gustafsson, Particle filter theory and practice with positioning applications, *IEEE Aerospace and Electronic Systems Magazine* 25 (7) (2010) 53–82, <https://doi.org/10.1109/maes.2010.5546308>.
- [10] S.F. Schmidt, Application of state-space methods to navigation problems, *Advances in Control Systems* 3 (1966) 293–340. Elsevier.
- [11] B.D. Tapley, B.E. Schutz, G.H. Born, *Statistical Orbit Determination*, 2004.
- [12] G.J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, Academic Press, 1977.
- [13] P.S. Maybeck, *Stochastic Models, Estimation, and Control*, Academic Press, 1979.
- [14] G. Xu, *GPS. Theory, Algorithms and Applications*, second ed., Springer, 2007.
- [15] J.A. Klobuchar, Ionospheric time-delay algorithm for single-frequency GPS users, *IEEE Transactions on Aerospace and Electronic Systems* (3) (1987) 325–331.
- [16] A. Hauschild, Basic Observation Equations, in: *Handbook of Global Navigation Satellite Systems*, in: P.J. Teunissen, O. Montenbruck (Eds.), *Springer Handbook of Global Navigation Satellite Systems*, vol 10, Springer International Publishing, New York, NY, USA, 2017, pp. 561–582.
- [17] J.F. Zumberge, M.B. Heflin, D.C. Jefferson, M.M. Watkins, F.H. Webb, Precise point positioning for the efficient and robust analysis of GPS data from large networks, *Journal of Geophysical Research: Solid Earth* 102 (B3) (1997) 5005–5017.
- [18] J.A. Christian, A tutorial on horizon-based optical navigation and attitude determination with space imaging systems, *IEEE Access* 9 (2021) 19819–19853.
- [19] S.I. Sheikh, D.J. Pines, P.S. Ray, K.S. Wood, M.N. Lovellette, M.T. Wolff, Spacecraft navigation using X-ray pulsars, *Journal of Guidance, Control, and Dynamics* 29 (2006) 49–63.

- [20] G. Wahba, A least squares estimate of satellite attitude, *SIAM Review* 7 (3) (1965), 409–409.
- [21] G.H. Golub, C.F. Van Loan, *Matrix Computations*, JHU Press, 2013.
- [22] J.E. Keat, Analysis of least-squares attitude determination routine DOAOP, Technical Report CSC/TM-77/6034, Computer Sciences Corporation (February 1977), <https://doi.org/10.5281/ZENODO.32291>.
- [23] M.D. Shuster, The QUEST for better attitudes, *Journal of the Astronautical Sciences* 54 (3) (2006) 657–683.
- [24] F.L. Markley, J.L. Crassidis, *Fundamentals of Spacecraft Attitude Determination and Control*, Space Technology Library, Springer, New York, 2014.
- [25] V. Pesce, M.F. Haydar, M. Lavagna, M. Lovera, Comparison of filtering techniques for relative attitude estimation of uncooperative space objects, *Aerospace Science and Technology* 84 (2019) 318–328.
- [26] F.L. Markley, Multiplicative vs. additive filtering for spacecraft attitude determination, *Dynamics and Control of Systems and Structures in Space* 467–474 (2004) 48.
- [27] E.J. Lefferts, F. Landis Markley, M.D. Shuster, Kalman filtering for spacecraft attitude estimation, *Journal of Guidance, Control, and Dynamics* 5 (5) (1982) 417–429.
- [28] R. Mahony, T. Hamel, J.-M. Pflimlin, Complementary filter design on the special orthogonal group SO (3), in: *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, 2005.
- [29] R. Mahony, T. Hamel, J.-M. Pflimlin, Nonlinear complementary filters on the special orthogonal group, *IEEE Transactions on Automatic Control* 53 (5) (2008) 1203–1218.
- [30] A. Colagrossi, M. Lavagna, Fault tolerant attitude and orbit determination system for small satellite platforms, *Aerospace* 9 (2022) 46, <https://doi.org/10.3390/aerospace9020046>.
- [31] R. Opronolla, G. Fasano, G. Rufino, M. Grassi, A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations, *Progress in Aerospace Sciences* 93 (2017) 53–72, <https://doi.org/10.1016/j.paerosci.2017.07.001>.
- [32] J.R. Carpenter, C.N. D’Souza, *Navigation Filter Best Practices*, 2018.
- [33] G.J. Awcock, T. Ray, *Applied Image Processing*, Macmillan International Higher Education, 1995.
- [34] N. Otsu, A threshold selection method from gray-level histograms, *IEEE transactions on Systems, Man, and Cybernetics* 9 (1) (1979) 62–66.
- [35] W. Niblack, *An Introduction to Digital Image Processing*, Strandberg Publishing Company, 1985.
- [36] H. Freeman, On the encoding of arbitrary geometric configurations, *IRE Transactions on Electronic Computers* 2 (1961) 260–268.
- [37] T.P. Wallace, P.A. Wintz, An efficient three-dimensional aircraft recognition algorithm using normalized Fourier descriptors, *Computer Graphics and Image Processing* 13 (2) (1980) 99–126.
- [38] T. Pavlidis, F. Ali, Computer recognition of handwritten numerals by polygonal approximations, *IEEE Transactions on Systems, Man, and Cybernetics* 6 (1975) 610–614.
- [39] A. Papoulis, S.U. Pillai, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, Boston, MA, 1991.
- [40] M. Savini, Moments in image analysis, *Alta Frequenza* 57 (2) (1988) 145–152.
- [41] P.W. Kitchin, A. Pugh, *Processing of binary images*, in: *Robot Vision*, Springer, Berlin, Heidelberg, 1983, pp. 21–42.
- [42] R. Gohane, S. Adatrao, M. Mittal, Comparison of various centroiding algorithms to determine the centroids of circular marks in images, in: *2016 International Conference*

- on Computing, Analytics and Security Trends (CAST), IEEE, December 2016, pp. 162–166.
- [43] J.A. Christian, Accurate planetary limb localization for image-based spacecraft navigation, *Journal of Spacecraft and Rockets* 54 (3) (2017) 708–730.
 - [44] W.H. Press, S.A. Keukolsky, W.T. Vettering, B.P. Flannery, Levenberg–Marquardt Method. *Numerical Recipes in C: The Art of Scientific Computation*, 1992, pp. 542–547.
 - [45] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Communications of the ACM* 24 (6) (1981) 381–395.
 - [46] R. Hartley, A. Zisserman, *Projective geometry and transformations of 2D. Multiple View Geometry in Computer Vision*, 2003.
 - [47] B. Triggs, P.F. McLauchlan, R.I. Hartley, A.W. Fitzgibbon, Bundle adjustment—a modern synthesis, in: *International Workshop on Vision Algorithms*, Springer, Berlin, Heidelberg, September 1999, pp. 298–372.
 - [48] M. Sonka, V. Hlavac, R. Boyle, *Image Processing, Analysis, and Machine Vision*, Cengage Learning, 2014.
 - [49] L.P. Cassinis, R. Fonod, E. Gill, Review of the robustness and applicability of monocular pose estimation systems for relative navigation with an uncooperative spacecraft, *Progress in Aerospace Sciences* 110 (2019) 100548.
 - [50] H.C. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, *Nature* 293 (5828) (1981) 133–135.
 - [51] D. Jiang, J. Yi, Comparison and study of classic feature point detection algorithm, in: *2012 International Conference on Computer Science and Service System*, IEEE, August 2012, pp. 2307–2309.
 - [52] S.B. Pollard, J.E. Mayhew, J.P. Frisby, PMF: a stereo correspondence algorithm using a disparity gradient limit, *Perception* 14 (4) (1985) 449–470.
 - [53] V. Pesce, R. Opronolla, S. Sarno, M. Lavagna, M. Grassi, Autonomous relative navigation around uncooperative spacecraft based on a single camera, *Aerospace Science and Technology* 84 (2019) 1070–1080.
 - [54] V. Pesce, M. Lavagna, R. Bevilacqua, Stereovision-based pose and inertia estimation of unknown and uncooperative space objects, *Advances in Space Research* 59 (1) (2017) 236–251.
 - [55] H. Durrant-Whyte, D. Rye, E. Nebot, Localization of autonomous guided vehicles, *Robotics Research* (1996) 613–625.
 - [56] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, J. Tardós, A comparison of loop closing techniques in monocular SLAM, *Robotics and Autonomous Systems* 57 (12) (2009) 1188–1197.



Control

Francesco Cavenago¹, Aureliano Rivolta², Emanuele Paolini²,
Francesco Sanfedino³, Andrea Colagrossi⁴, Stefano Silvestrini⁴,
Vincenzo Pesce⁵

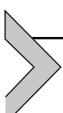
¹Leonardo, Milan, Italy

²D-Orbit, Fino Mornasco, Italy

³ISAE-SUPAERO, Toulouse, France

⁴Politecnico di Milano, Milan, Italy

⁵Airbus D&S Advanced Studies, Toulouse, France



What is control?

Control is the last word closing the GNC acronym, as it is the last conceptual block in the guidance, navigation, and control loop. Let us consider again our person going outside the shelter, planning the way and acknowledging the current location, aiming to reach the final destination. This task can be completed if the person starts walking and maintains the actual path on the desired one. Thus, the individual shall control the motion commanding actions to legs and arms. The body will start moving maintaining the upright posture and interacting with the external environment. The latter influences the real movements, as a wind from the back may result in longer steps for a given push on the feet or, on the contrary, walking uphill requires stronger commands to the leg to maintain the desired path and pace. This is natural for everyone, but it is a complex task involving the deviation of the navigation estimation of the current state with respect to the guidance plan. Moreover, it entails the computation of the control actions to track the desired path and recover possible drifts, calculating the commands to apply forces and torques on the body dynamics. The example of a walking person allowed also to have a simple visualization of the main functions related with *control*.

In the same example, an additional control loop is present, despite it is less evident. In fact, maintaining the upright posture, the person uses different state measurement and commands other actions to preserve the orientation of the body. In this case, the desired state is not a trajectory, but a single reference point. This GNC loop is dedicated to keep a static orientation state, over a dynamic reference tracking. The whole human

body dynamics is obviously coupled, but the ability to separate the two control effects is based on the higher frequency of the equilibrium control loop with respect to the translational one. This is not an easy task, but the concepts to master multiple degrees of freedom control will be learnt across this chapter, since it is extremely relevant for a spacecraft traveling across the space, as it is for a person exploring the Earth (just remembering the rocking motion of our first walking experiences).

Control is not only the computation of actions to reduce the errors with respect to a certain number of desired states but it is also the correct evaluation of the commands to be applied to generate forces and torques. Even if the formers are the main tasks of what is generally defined a *controller*, and the latter are those associated with an *actuator*, the proper control design shall consider both. In fact, control functions shall contain specific instructions, or actuation functions, to translate and send the control commands to the actuators. For example, imagine the modifications to apply in the control block if our walking person wants to reach the destination by riding a bicycle or driving a car. Furthermore, the specifications of the actuators affect the controller design, such as, among the others, the maximum intensity of the computed control actions shall be within the limits of the actuators: our person cannot run faster than the training allows.

Control has to command to the spacecraft's actuators the actions to maintain the current state close to the desired one. Clearly, if the control action is not properly computed by the control algorithms, or if it is badly actuated, the discrepancy between the navigation estimated state and the guidance reference trajectory will tend to increase.

With these notions in mind, we have the basics to start exploring the technicalities of control algorithms and functions.

This chapter is composed by the following sections:

- *Control design.* In this section, a comprehensive description of the control design process is given. It includes the control design process both for the state space and frequency domain. Furthermore, an introduction to nonlinear control design is presented.
- *Review of control methods.* This section presents a review of the most popular control techniques, from Proportional-Integral-Derivative (PID) control to Linear Quadratic Regulator (LQR). The content covers also adaptive and robust control basics, and it includes Model Predictive Control (MPC) and Sliding Mode Control (SMC).
- *Control budget.* This section covers the typical budgets to account for during control system design, with reference to space standards.

- *Control implementation best practice.* A series of best practice tips and checks is collected in this section, giving a more practical perspective on the implementation of control algorithms.



Control design

In this section, some of the main methods and techniques to address a control problem are reviewed. In the first part, the properties of feedback and the general objective of the control are presented. Then, the attention is focused on linear time-invariant (LTI) systems, and the design in the frequency domain and in the state space is discussed. Finally, some tools are introduced to deal with nonlinear systems. Note that this chapter is not meant to cover exhaustively all the mentioned control topics, but it provides an overview and the basis to use it. The reader is invited to refer to the specialized books, cited along the text, for a further deepening of the subject.

Basic terminology

This short paragraph provides the readers the terminology and basic notions necessary to understand the concepts reported in this chapter. Further details can be found in the Appendix—A2.

- *Plant.* The system to be controlled will be often called plant. It can be modeled in the time domain, i.e., as function of the time variable t , using the state space representation. If the plant is LTI, it can also be modeled in the frequency domain, i.e., as function of the generalized frequency s (complex variable), through the mathematical framework of the transfer function. Time invariance means that the system is not a direct function of the time, namely t does not appear explicitly in the equations.
- *State.* The state of a system is a set of variables which are able to represent the condition of the system. Known the values of these variables at a certain time instant and the time evolution of the input, the dynamic behavior of the system can be determined. In a mechanical system, position and velocity are typically used as state variables.
- *Transfer function.* Let us consider an LTI system and imagine it as a process, whose output is affected by some input. The transfer function is the mathematical mean to describe the relation between the input and the output. It is usually determined as the ratio between the Laplace transform of the output and the input with initial condition zero, where the Laplace transform is an integral transform used to pass from the

time domain to the frequency domain, i.e., from the time variable t to the generalized frequency s .

- *Zeros and poles.* When a system can be modeled as a rational transfer function, the roots (generally complex number) of the numerator and denominator are called zeros and poles, respectively. The dynamic behavior of the system is strictly related to these quantities. In particular, as it will be shown later, the stability of the system depends on the position of the poles in the complex plane.
- *SISO.* The acronym SISO stands for single-input single-output. Therefore, a SISO system is a process with only one input and one output.
- *MIMO.* The acronym MIMO stands for multi-input multi-output. Therefore, a MIMO system is a process showing more than one input and output.

Properties of feedback control

Spacecraft control can be achieved by implementing either open-loop or closed-loop (negative feedback) schemes. Looking at the block diagram representation in Fig. 10.1, it can be noted that, in the open-loop schemes (a), the control action u is independent of the actual response of the system y and does not rely on any measurements. The maneuver is predetermined, and the control action is usually computed solving an optimal control problem. This procedure is performed on ground and then the control commands are sent to the spacecraft for the execution. Once determined, the control is not affected by the actual behavior of the system, and consequently the open-

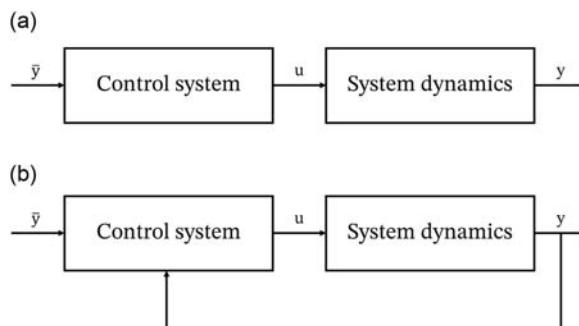


Figure 10.1 Open- and closed-loop schemes, where \bar{y} is the reference signal, u is the control action, and y is the system response. (a) Open-loop scheme. (b) Closed-loop scheme.

loop strategies result to be quite sensitive to uncertainty in the model parameters and disturbances. The activities performed to generate off-line solutions for the control profile fall into the definition of guidance, as discussed in the previous chapter. A good knowledge of the system and the environment is necessary to achieve satisfactory performance.

Conversely, in feedback schemes (Fig. 10.1b), the controlled variable is measured, and the control command is computed based on the difference between its value and a reference one. Basically, feedback control is a reactive strategy: whenever a deviation from the desired behavior occurs, a corrective action is exerted on the system. This results in a very interesting robust performance in the presence of model uncertainty and unknown disturbances. An error in the regulated signal introduced by these nonidealities can be sensed and compensated (or partially compensated) by the control. This is one of the key properties of closed-loop strategies and one of the reasons why they are widely used for spacecraft control.

Another important use of feedback is to modify the dynamics of the controlled system in such a way to satisfy the application needs. For example, the responsiveness and the response damping of the plant can be changed through the closure of a feedback loop and a proper tuning of the control parameters. An unstable system can even be stabilized. Thanks to this property, it is possible to shape the behavior of a system (although with some limitations which will be discussed in the following) to achieve the desired performance. Nevertheless, one must bear in mind that there is also the other side of the coin. Indeed, when two or more systems are interconnected in a cycle through feedback, their dynamics is strongly coupled, i.e., each system influences the others. The closed-loop behavior can become counterintuitive, difficult to predict, and, potentially, instabilities can arise. Moreover, feedback injects measurement noise into the system, which can lead to undesired effects if not treated properly (e.g., through filtering). For this reason, formal methods and tools are necessary to design and analyze a feedback control, and some of them will be presented in the following paragraphs.

Control objective and performance

An important step toward a well-designed control system is the definition of the performance required to accomplish the task. Consider the general control scheme reported in the block diagram of Fig. 10.2. The variables \bar{y} and y are the reference and controlled signals, respectively, and u is the control

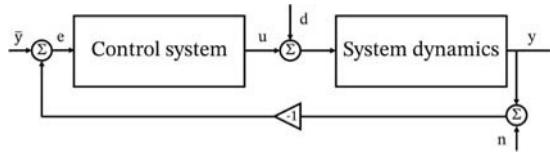


Figure 10.2 General closed-loop system, where \bar{y} is the reference signal, e is the control error, u is the control action, y is the system response, d is the load disturbance, and n is the noise measurement.

command. Finally, d and n are the disturbance acting on the system and the noise on the measurements, respectively. Basically, the objective of the control is to achieve $y = \bar{y}$ rejecting disturbance and noise and limiting the control action. Note that the reference can be constant or vary in time. The former case is called regulation control, while the latter one is the tracking control. This general objective can be expressed in a more formal way by defining some requirements and performance the control system shall meet:

- Closed-loop stability
- Static and dynamic performance: steady-state error, rise time, settling time, overshoot
- Disturbance attenuation
- Measurement noise filtering
- Robustness to uncertainty

Each property is now briefly discussed.

Closed-loop stability

Consider a solution of a dynamic system $\mathbf{x}_n(t; \mathbf{x}_{n,0})$ with initial condition $\mathbf{x}_{n,0}$ and imagine to perturb the initial condition $\mathbf{x}_{p,0}$. The nominal solution $\mathbf{x}_n(t; \mathbf{x}_{n,0})$ is stable if the perturbed one, $\mathbf{x}_p(t; \mathbf{x}_{p,0})$, stays close to it for all the time instants. In a more formal way, $\mathbf{x}_n(t; \mathbf{x}_{n,0})$ is stable if for all $\epsilon > 0$, there exists a $\delta > 0$ such that:

$$\|\mathbf{x}_{n,0} - \mathbf{x}_{p,0}\| < \delta \rightarrow \|\mathbf{x}_n(t; \mathbf{x}_{n,0}) - \mathbf{x}_p(t; \mathbf{x}_{p,0})\| < \epsilon \text{ for all } t > 0. \quad (10.1)$$

Clearly, a solution that is not stable is unstable, i.e., a perturbed solution moves away from the nominal one indefinitely. A special case is when the nominal solution is an equilibrium solution. In this circumstance, it is simply said that the equilibrium point is stable.

If Eq. (10.1) is satisfied, and the following condition is also verified:

$$\lim_{t \rightarrow \infty} \|\mathbf{x}_n(t; \mathbf{x}_{n,0}) - \mathbf{x}_p(t; \mathbf{x}_{p,0})\| = 0, \quad \forall \mathbf{x}_{p,0}: \|\mathbf{x}_{n,0} - \mathbf{x}_{p,0}\| < \delta, \quad (10.2)$$

The solution is said to be asymptotically stable. This additional condition means that a perturbed solution converges to the nominal one as t approaches infinity for a $\mathbf{x}_{p,0}$ sufficiently close to $\mathbf{x}_{n,0}$. Depending on the domain of applicability, a solution can be locally asymptotically stable or globally asymptotically stable. In the former case, a perturbed solution is attracted to the nominal one only if $\mathbf{x}_{p,0}$ lies in a limited domain around $\mathbf{x}_{n,0}$, while, in the latter one, it converges for any initial condition.

When a feedback loop is closed, the stability of the system must be analyzed and verified. It will be shown that, for an LTI closed-loop system, it is sufficient to study the closed-loop poles. For the asymptotic stability, all the poles have to lie in the left-half complex plane, i.e., the real part must be negative. In this case, the stability is a property of the system. Indeed, it is possible to talk about the stability of the system rather than the stability of a solution or of an equilibrium point. In case of nonlinear system, Lyapunov theorem will be introduced to analyze the stability of an equilibrium point.

Static and dynamic performance

The capability of the closed-loop system to follow a reference signal is usually expressed in terms of steady-state error (static performance) and transient behavior (dynamic performance). The main performance measures are represented in Fig. 10.3.

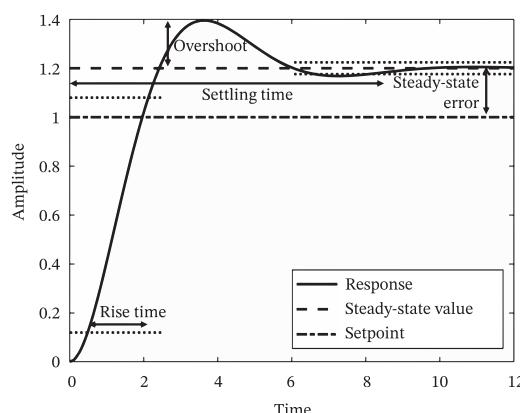


Figure 10.3 Time performance representation.

The steady-state error is the error between the reference and the controlled signal after the initial transient. During the design phase, it is usually evaluated looking at the response to some test reference signals, such as step input and ramp input. Ideally, the steady-state error should be zero. However, in practice, it never happens due to noise in the measurements, quantization, uncertainty in the model, and disturbances. Typically, the specific application requires a certain level of final accuracy which can be specified as a maximum steady-state error.

During the transient, the controlled variable passes from one steady state to another, and its dynamic behavior must satisfy some responsiveness and damping requirements. Generally, the responsiveness of the system is evaluated in terms of rise time, which is the amount of time required by the output to go from 10% to 90% of its final value. The damping of the closed-loop response is expressed in terms of overshoot and settling time. The former one quantifies the maximum deviation of the controlled variable from its steady-state value, and it is usually defined as a percentage of the final value. Note that it is assumed that, in the subsequent time instants, the variable does not exceed the steady-state value more than this initial transient. The settling time is the amount of time required by the signal to stay within some percentage of the final value (typically, 1%, 2%, or 5%). This dynamic performance is commonly evaluated looking at the response of the closed-loop system to a step input. In general, this performance depends on the size of the step, except for the linear system, for which the dynamic performance is independent of it.

Disturbance and measurement noise rejection

Disturbance and measurement noise rejection expresses the capability of the closed-loop system to attenuate the effects of these perturbations on the response, and thus to obtain satisfactory tracking performance. It will be shown that guaranteeing these properties means to impose a certain shape to the system transfer function, which describes the relation between the reference and the output variable. In particular, the so-called open-loop transfer function (i.e., the transfer function between the reference and controlled variable opening the control loop) must have high gain at low frequency to attenuate disturbance on the plant and low gain at high frequency to filter noise on the feedback line. This aspect will be discussed more in detail later in this chapter.

Robustness to uncertainty

A well-designed control system must ensure that closed-loop system behaves satisfactorily not only in the nominal condition but also in the presence of model uncertainty and/or variation of some parameters. This means that, even in off-nominal scenarios, the stability must be guaranteed, and the static and dynamic performance must not deviate excessively from the nominal one. Clearly, the closed-loop system cannot be robust to any perturbation. The control designer should evaluate and quantify the expected uncertainty and parameters variation and perform analyses to assess the behavior of the closed-loop system for these perturbations. It is possible to define some stability margins, which provide good indications about the robustness of the closed-loop system, even if they cannot guarantee it. They will be presented in section [Chapter 10 – Control – Stability and stability margin](#).

Controllability

As declared previously, the objective of the control is to modify the behavior of a system in such a way to obtain the desired response. To achieve this goal, the control input must be able to influence the state of the system, but to what extent is this possible? To answer this question, controllability of the system must be analyzed. A system is said to be controllable if there exists a suitable control signal that can move the state from any initial condition to any final condition in a finite time. How is this property verified? Consider a general linear time-invariant system in state-space formulation:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}, \quad (10.3)$$

where \mathbf{x} is the $n \times 1$ vector of the state, \mathbf{u} is the $m \times 1$ vector of the input, \mathbf{A} is the $n \times n$ dynamics matrix, and \mathbf{B} is the $n \times m$ control matrix. The state Eq. (10.3) is controllable if the $n \times m$ controllability matrix:

$$\mathbf{W} = [\mathbf{B} \quad \mathbf{AB} \quad \mathbf{A}^2\mathbf{B} \quad \cdots \quad \mathbf{A}^{n-1}\mathbf{B}] \quad (10.4)$$

has rank equal to n . In the following, it will be shown that there is an important connection between the controllability and the design of state feedback control law. In particular, if a system is controllable, it is possible to modify via feedback all the eigenvalues of the dynamics matrix \mathbf{A} , and thus to shape the transient behavior of the system according to the needs.

Control design in frequency domain

The first techniques to design a controller, that are presented, are developed in the frequency domain. Frequency domain modeling is a powerful tool to analyze the closed-loop behavior of an LTI system. It provides significant insights into the properties of the system, such as stability, robustness, and attenuation of disturbances. Having in mind the general feedback scheme in Fig. 10.2, the idea is to represent the system in terms of input–output relations between the variables, and then study their frequency response. To this aim, Laplace transform is used to move from the time domain in the variable t to the frequency domain in the variable s , which is called generalized frequency. Afterward, the input–output relation is described by the so-called transfer function, which is defined as the ratio of the Laplace transforms of the output and the input when the initial state is zero. From the analysis of the transfer functions, it is possible to investigate the stability of the system, and understand how the controlled variable, the error, and the control action (i.e., output) are affected by the reference signal, the disturbances, and the noise (i.e., input). In the following, methods and techniques are presented for SISO systems, for which initially were developed. The review of the concepts reported in this section and the notation is mainly based on Refs. [1,2], which are excellent reference books for the reader who wants to deepen the control design based on the transfer functions.

Stability and stability margins

The primary requirement a control designer has to satisfy is the stability, which is typically verified by the analysis of the roots of the characteristic polynomial. With reference to Fig. 10.4, where $C(s)$ is the transfer function of the controller and $P(s)$ is the transfer function of the controlled plant, the relation between the Laplace transforms of the reference $\bar{Y}(s)$ and the output variable $Y(s)$, i.e., the closed-loop transfer function, is expressed as:

$$\frac{\bar{Y}}{Y} = \frac{PC}{1 + PC} = \frac{L}{1 + L} = \frac{N_L}{N_L + D_L}, \quad (10.5)$$

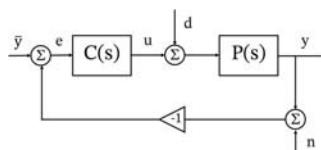


Figure 10.4 General closed-loop system, where $C(s)$ is the controller transfer function, $P(s)$ is the plant transfer function, \bar{y} is the reference signal, e is the control error, u is the control action, y is the system response, d is the load disturbance, and n is the noise measurement.

where $L = PC$ is the so-called open-loop transfer function, N_L and D_L are the numerator and the denominator of L , respectively. Hereafter, the dependance on the generalized frequency s is omitted for simplicity. The characteristic polynomial of the closed-loop system is computed as follows:

$$\lambda = N_L + D_L. \quad (10.6)$$

The closed-loop system is asymptotically stable if all the roots of the characteristic polynomial, also called poles of the system, are in the left half-plane (L.H.P.) of the complex plane, namely they have negative real part. The poles correspond to the eigenvalues of the state-space dynamics matrix \mathbf{A} .

The analysis of the characteristic polynomial is quite straightforward and says if the controlled system is stable or not. However, a designer is interested in collecting more information on the stability which can provide a useful guide for the design of the control, like how far the system is from instability or how much it is robust to perturbations. Basically, it is important to define some margins of the stability. To this aim, a key result is the Nyquist stability criterion, which is briefly recalled here as it is most often used in practice, and thus without going into the more formal definition and all the theoretical details. For a more complete discussion, refer to Ref. [1].

The Nyquist criterion is based on the Nyquist plot of the loop transfer function $L = PC$. It is recalled that the Nyquist plot is a polar diagram in the complex plane representing the frequency response of a transfer function. The curve is traced from $\omega = 0$ to $\omega = +\infty$, with ω being the frequency, and then the part symmetric to the real axis is added. Denoting by P_d the number of poles of L in the right half-plane (R.H.P.) and by N the number of net encirclements of the diagram around the -1 point of the real axis (counted positive counterclockwise), the Nyquist criterion states that *the closed-loop system is asymptotically stable if $P_d = N$ (winding number condition) and the diagram does not pass through the -1 point (critical point).*

The Nyquist criterion requires that the curve does not intersect the real axis in the critical point. What if it happens? A very simple test example is proposed to answer this question. An open-loop transfer function L is considered, which results in a closed-loop transfer function with two dominant complex conjugate poles in the L.H.P. close to the imaginary axis. The Nyquist curve of L , the dominant complex conjugate poles (cross markers), and step response of the closed-loop system are reported in Fig. 10.5a. The transfer function is then multiplied by a factor 3, and the resulting plots are reported in Fig. 10.5b. Finally, the initial transfer function is multiplied by a

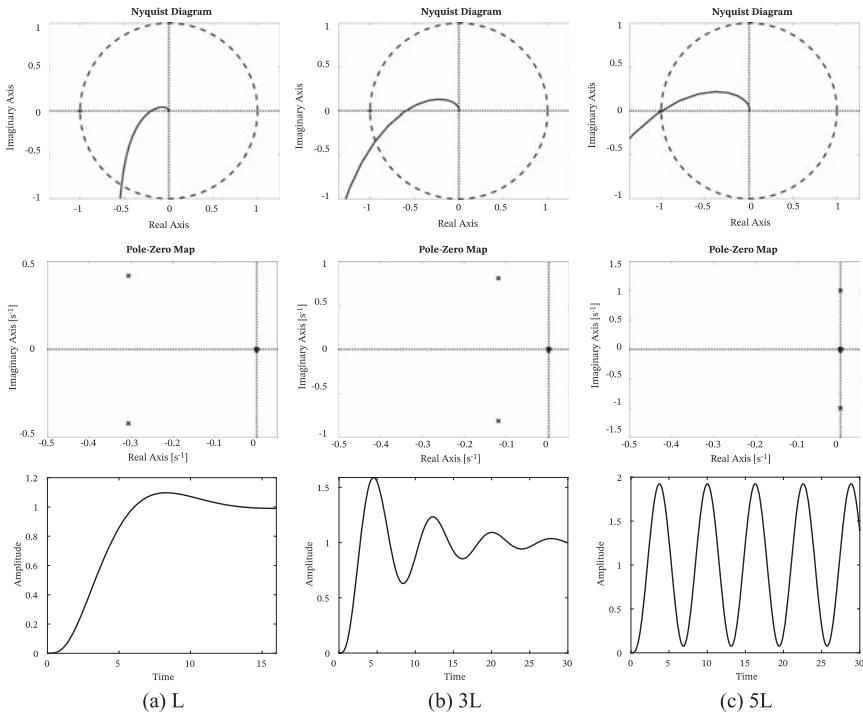


Figure 10.5 Behavior of the system when the Nyquist curve approaches the critical point. (a) Nyquist curve of the transfer function L , dominant poles, and time response of the closed-loop system. (b) Nyquist curve of the transfer function $3L$, dominant poles, and time response of the closed-loop system. (c) Nyquist curve of the transfer function $5L$, dominant poles, and time response of the closed-loop system.

factor 5, and the associated plot are in Fig. 10.5c. As the Nyquist curve approaches the critical point, the dominant complex poles approach the imaginary axis, and the time response shows increased oscillation. When the curve passes exactly through the -1 point, the complex poles lie on the imaginary axis, and, in this case, the time response shows a pure oscillatory behavior. Potentially, the system could become even unstable depending on the multiplicity of the poles at the origin.

Therefore, one of the key messages of the Nyquist theorem is to avoid the critical point -1 . Some measures can be introduced to characterize how close the Nyquist curve is to this point, and thus to express the degree of stability of the system. The most obvious one is the direct computation of the shortest distance between the curve and the critical point (see Fig. 10.6).

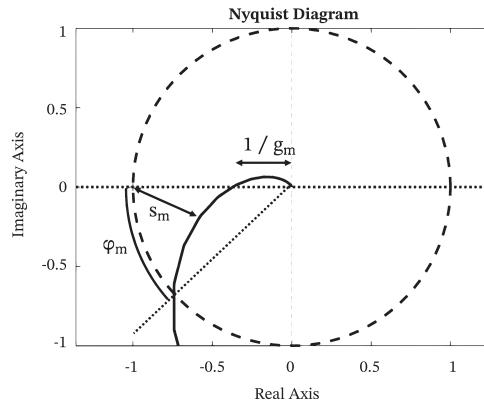


Figure 10.6 Graphical illustration of the stability margin s_m , gain margin g_m , and phase margin ϕ_m .

This distance is called stability margin or modulus margin and is computed as follows:

$$s_m = \min_{\omega} (||1 + L(j\omega)||), \quad (10.7)$$

or, in decibel, as:

$$s_m = \max_{\omega} (-20 \log_{10} (||1 + L(j\omega)||)). \quad (10.8)$$

Other margins can also be defined based on the observation that an increase of the controller gain expands the Nyquist curve radially, and an increase of the phase turns it clockwise. One might wonder what is the amount of gain or phase that makes the Nyquist curve pass through the critical point, and consequently the system unstable. The answer to this question defines the so-called gain margin and phase margin.

The gain margin is defined as the smallest gain causing instability that can be provided to the closed-loop system. It can be derived from the Nyquist plot as the inverse of the distance between the origin and the intersection of the curve with the negative real axis between 0 and -1 (Fig. 10.6). In case of multiple intersections, the closest to the critical point is considered. The intersection can be found evaluating the gain of the loop transfer function at the so-called phase crossover frequency, i.e., the frequency at which

the phase is -180° . Therefore, denoting by g_m the gain margin, it is computed as follows:

$$g_m = \frac{1}{\left| L(j\omega_{pc}) \right|}, \quad (10.9)$$

where ω_{pc} is the phase crossover frequency. As for the modulus margin, the gain margin is sometimes computed in decibel as $g_m = -20 \log_{10}(\left| L(j\omega_{pc}) \right|)$.

The phase margin can be visualized in the Nyquist plot as the angle between the negative real axis and the line passing through the origin and the intersection between the curve and the unit half-circle below the real axis (Fig. 10.6). It is the amount of phase necessary to reach the stability limit. Also in this case, if the curve intersects the half-circle multiple times, the intersection closest to the critical point is considered. In mathematical terms, the phase margin ϕ_m is defined as:

$$\phi_m = 180^\circ + \arg(L(j\omega_{gc})), \quad (10.10)$$

where ω_{gc} is the gain crossover frequency, namely the frequency at which the modulus of the loop transfer function is equal to 1 (or 0 dB).

Typically, the use of the gain and phase margins is more widespread because they have a clear representation in the Bode plot of the loop transfer function (Fig. 10.7), which has been a popular tool for the control design. Indeed, from the Bode plot, it is easy to see at what frequency the phase is -180° , and then, the gain margin is the inverse of the gain at that frequency. Similarly, it is possible to identify the frequency at which the loop transfer function crosses the 0 dB line, and the phase margin is simply the phase at that frequency plus 180° . However, the reader should keep in mind that gain and phase margins could not guarantee the robustness of the system. Situations can arise in which although these margins are satisfied, the response of the system shows undesired oscillations. For this reason, it is best practice to define the robustness of the closed-loop through all the three margins, i.e., gain, phase, and modulus margins.

Generally, the system considered for the control design is not exactly the actual system due to the presence of uncertainties and effects that are not easy to model. Taking as example the specific case of a satellite, consider the dynamic couplings due to flexible appendages or sloshing of fuel mass. These perturbations/unmodeled dynamics can cause significant deviations from

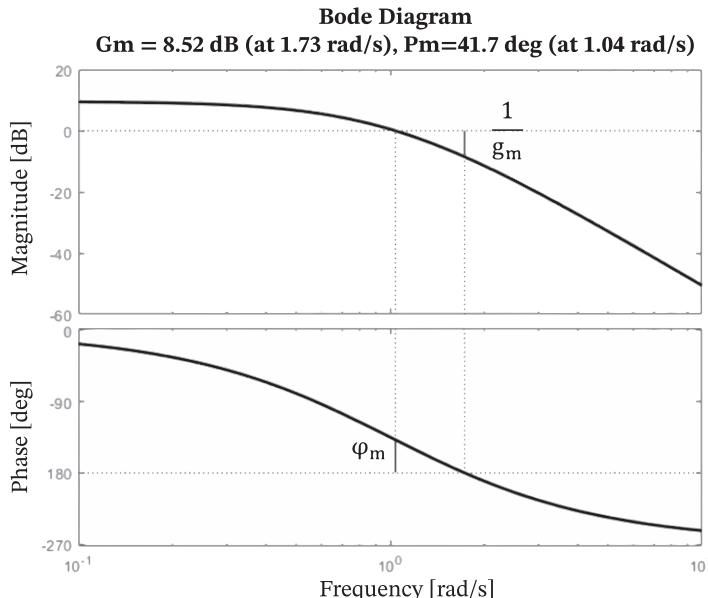


Figure 10.7 Representation of the gain margin g_m and phase margin ϕ_m on the Bode plot.

the nominal behavior. Moreover, a satellite is usually required to operate in different conditions along the entire mission (variation of mass and inertia properties). Consequently, providing sufficiently high margins is essential for a robust control design. Recommended values in the European Cooperation for Space Standardization (ECSS) standards [3,4] are phase margin $\phi_m \geq 30^\circ$, gain margin $g_m \geq 2$ (in decibel, ≥ 6 dB), and modulus margin $s_m \geq 0.5$ (in decibel, ≥ -6 dB).

Sometimes, another margin is defined and specified, which is called delay margin. It quantifies the minimum delay that makes the controlled system to become unstable. The delay margin is computed as follows:

$$d_m = \frac{180^\circ + \arg(L(j\omega_{gc}))}{\omega_{gc}} \frac{\pi}{180^\circ}. \quad (10.11)$$

Sensitivity functions

The analysis in the frequency domain is based on the study of the relation between the input and the output of a system, which is well-described by mathematical framework of the transfer function. Considering the block

diagram of the closed-loop system in Fig. 10.4, the reference signal \bar{y} , the load disturbance d , and the measurement noise n can be defined as the external input to the system, while the controlled variable y , the error e , and the control action u are the output. The transfer functions describing the relations among them are reported in Table 10.1.

It can be noted that some transfer functions in Table 10.1 are repeated more than once. Basically, most of the time, the frequency analysis of a closed-loop system focuses on a subset, which is known as the *Gang of four* [1]:

$$S = \frac{1}{1 + PC} = \frac{1}{1 + L} \quad T = \frac{PC}{1 + PC} = \frac{L}{1 + L} \quad (10.12)$$

$$PS = \frac{P}{1 + PC} = \frac{P}{1 + L} \quad CS = \frac{C}{1 + PC} = \frac{C}{1 + L}$$

The transfer functions S and T are called *sensitivity function* and *complementary sensitivity function*, respectively, while PS and CS are the *load sensitivity function* and *noise sensitivity function*. These four functions are particularly useful to study and understand the performance of the closed-loop system. The closed-loop response y and the control error ζ can be related to the sensitivity functions as follows:

$$y = T\bar{y} + PSd - TN, \quad (10.13)$$

$$\zeta = \bar{y} - y = (1 - T)\bar{y} - PSd + TN. \quad (10.14)$$

Analyzing Eqs. (10.13) and (10.14), some reasoning can be made. First, a comparison between Eq. (10.13) and the open-loop response, $y = PC\bar{y} + Pd$, highlights some of the properties of feedback control discussed in section Chapter 10 – Control – Properties of feedback control. It can be noted that the open-loop control has no effect on the disturbance which acts on the system without attenuation. Conversely, in Eq. (10.13), the disturbance is multiplied by the sensitivity function which can be properly shaped for good rejection. A drawback of feedback scheme is the injection of

Table 10.1 Transfer functions of the closed-loop system in Fig. 10.4.

	y	e	u
\bar{y}	$\frac{PC}{1+PC}$	$\frac{1}{1+PC}$	$\frac{C}{1+PC}$
d	$\frac{P}{1+PC}$	$\frac{-P}{1+PC}$	$\frac{-PC}{1+PC}$
n	$\frac{-PC}{1+PC}$	$\frac{-1}{1+PC}$	$\frac{-C}{1+PC}$

measurement noise in the system. As regards the response to the setpoint, both control schemes can be used to improve it. It will be shown later that feedback and feedforward control can be combined to achieve better performance of the system: feedback is typically used to increase robustness and attenuate disturbance, while feedforward term is added for better tracking performance. A second reasoning starts from Eq. (10.14). Ideally, the controller should achieve $\zeta = 0$. Focusing on the first two terms of the equation, i.e., setpoint and disturbance responses, a zero control error means shaping the sensitivity function and the complementary sensitivity function in such a way to have $T \approx 1$ and $S \approx 0$. It is recalled that $S + T = 1$, and thus these two conditions agree and can be met in the same frequency range. However, when the analysis is expanded including the noise, a proper rejection of this disturbance implies $T \approx 0$ (or equivalently $S \approx 1$). This simple analysis reveals the main feature of feedback design: a trade-off between opposite control objectives must be always performed. Fortunately, an application usually requires satisfying the control objectives over different frequency range. For instance, most of the time, the designer is interested in having good tracking and disturbance rejection performance at low frequency and noise attenuation at high frequency. How the sensitivity functions should be shaped to achieve these goals is explained in more detail in the following paragraphs.

Before proceeding further, the internal stability of the system is briefly discussed. In the previous paragraph, it has been shown that the stability can be studied looking at the poles of the closed-loop transfer function (i.e., T). Note that, since the transfer functions are often polynomials in s , the numerator and denominator could have a common factor, which is canceled. For example, when the loop transfer function L is computed as the product of P and C , pole/zero cancellation can occur. Sometimes this is just an algebraic simplification, but there are also situations in which this cancellation hides potential issues of the system. If the pole/zero cancellation involves an unstable pole, it can happen that $1 + L$ has no unstable roots, but one of the sensitivity functions might be unstable. Consequently, it is clear the importance to verify the stability of all the transfer functions of the system. If all the transfer functions are stable, the system is said to be *internally stable*. Another recommendation that can be derived is to avoid cancellation of unstable poles when designing the controller.

Response to setpoint

The response of the system to the reference signal is captured by the complementary sensitivity function T . In order to provide zero control error, it

has been noted that T should be ≈ 1 , but this cannot be achieved over the entire frequency range. Generally, an application requires good tracking performance up to a certain frequency which defines the bandwidth of the controller, denoted by ω_b . Inside the bandwidth, the relation $T \approx 1$ is verified (at the frequency identifying the bandwidth, $\|T\| = \frac{1}{\sqrt{2}}$ (-3 dB)).

Typically, the complementary sensitivity function takes the shape of a low-pass filter with unit zero frequency gain. Note that satisfying the condition $T \approx 1$ can be seen as a requirement for the loop transfer function L . Indeed, it is equivalent to ask for a gain of L as large as possible.

For some applications, it could be necessary to have a steady-state tracking error equal to zero (at least ideally). To this aim, the rule is that L shall have at least one integrator for each integrator of the reference signal. Imagine having $L = \tilde{L}/s^{n_L}$, with $\tilde{L}(0) \neq 0$ and n_L an integer number ≥ 1 , and a setpoint $\bar{y} = 1/s^{n_{\bar{y}}}$, with $n_{\bar{y}}$ an integer number ≥ 1 , using the final value theorem for Laplace transforms [2]:

$$\lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} s e(s) = \lim_{s \rightarrow 0} s \left(1 - \frac{L}{1 + L} \right) \bar{y} = \lim_{s \rightarrow 0} s \frac{s^{n_L - n_{\bar{y}}}}{s^{n_L} + \tilde{L}} \quad (10.15)$$

The tracking error is zero only if $n_L \geq n_{\bar{y}}$.

Some important parameters of the complementary sensitivity function are the bandwidth and the resonant peak value, which is computed as $M_T = \max_{\omega} \|T(j\omega)\|$ (or $M_T = \|T(j\omega)\|_{\infty}$ in terms of infinity norm).

Sometimes, specifications are expressed in terms of these characteristics because they are strictly related to the performance in the time domain. Indeed, larger bandwidth results in a faster response to setpoint, and the resonant peak value provides a measure of the overshoot, i.e., higher the resonant peak in frequency domain, larger the overshoot in the time domain. To get some further insight into the relation between the time and the frequency response, consider as example the following complementary sensitivity function:

$$T = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}, \quad (10.16)$$

which represents a second-order system. The parameter ω_n is the natural frequency and ξ is the damping. For a second-order system, it is possible to

write some relations between these parameters and the characteristics of the response in the time domain (to a step input) and in the frequency domain. These relations are reported in [Table 10.2](#). The variable ψ is computed as $\psi = \cos^{-1}\xi$.

An increase of the bandwidth means an increase of the natural frequency (given ξ), which results in a decrease of the rise and settling time. An increase of the damping means a decrease of the resonant peak, and consequently a limited overshoot. A graphical illustration is given in [Fig. 10.8](#).

Disturbance rejection

It has been shown that feedback loop introduces an attenuation of the disturbance acting on the system through the sensitivity function S . To this aim, the gain of S should be as low as possible over the expected frequency range of the disturbance. Indeed, the attenuation is effective only in the frequency range in which the gain of S is lower than 1. On the other hand, disturbances with frequency over which the gain of S is greater than 1 are amplified. The (lowest) frequency at which $\|S\| = 1$ is a parameter of the sensitivity function, and it is called sensitivity crossover frequency, ω_{sc} . Bandwidth, gain crossover frequency, and sensitivity crossover frequency are related to each other, and usually $\omega_{sc} \leq \omega_{gc} \leq \omega_b$ (for system with $\phi_m = 60$ deg, they all coincide). Typically, the controller is required to reject disturbances that are at low frequency, and thus the requirement of $S \approx 0$ at low frequency is in accordance with the requirement of $T \approx 1$ for good tracking performance since $T + S = 1$.

Another important parameter, which characterizes the system response to disturbance, is the peak of the sensitivity function, M_s . This value provides a measure of the maximum amplification of the disturbance. It is computed as $M_s = \max_{\omega} \|S(j\omega)\|$ (or $M_s = \|S(j\omega)\|_{\infty}$ in terms of infinity norm). Note

Table 10.2 Properties of the step response and frequency response for a second-order system.

Time domain	Frequency domain
$T_{rise} = \frac{\psi}{\omega_n}$	$\omega_b = \omega_n \sqrt{1 - 2\xi^2 \sqrt{(1 - 2\xi^2)^2 + 1}}$
$S_{overshoot} = e^{-\frac{\pi\xi}{\sqrt{1-\xi^2}}}$	$M_T = \begin{cases} 1/(2\xi\sqrt{1-\xi^2}), & \xi \leq \sqrt{2}/2 \\ N/A, & \xi > \sqrt{2}/2 \end{cases}$
$T_{settling} \approx \frac{4.6}{\xi\omega_n}$	

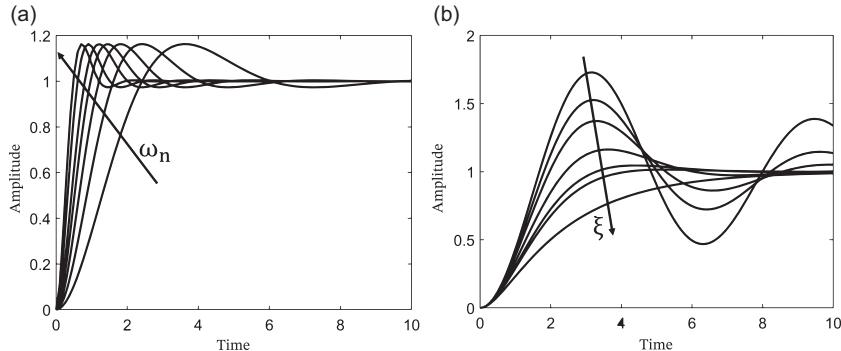


Figure 10.8 Second-order step response to vary natural frequency (left) and damping (right).

that looking for the maximum of $\frac{1}{||1+L(j\omega)||}$ corresponds to searching the minimum of $||1 + L(j\omega)||$, which is exactly the stability modulus. Consequently, it is possible to write $M_s = \frac{1}{s_m}$, which reveals that the peak of the sensitivity function is also a measure of robustness of the system. Generally, reasonable values for M_s should be lower than 2 (or 6 dB).

Observing that low gain of the sensitivity function means high gain of the loop transfer function L , the transfer function PS , from d to y , can be simplified at low frequency as:

$$PS = \frac{P}{1 + L} \approx \frac{P}{L} = \frac{1}{C}. \quad (10.17)$$

From Eq. (10.17), it is evident that high gains of the controller are beneficial for the disturbance attenuation. Often, an integrator is added to the controller to improve the performance. Indeed, the integral action guarantees that the steady-state error due to a constant disturbance goes to zero. This can be easily seen applying the final value theorem to a step response:

$$\lim_{t \rightarrow \infty} e_d(t) = \lim_{s \rightarrow 0} sPS \frac{1}{s} = \lim_{s \rightarrow 0} s \frac{T}{C} \frac{1}{s} = \lim_{s \rightarrow 0} \frac{T}{C}, \quad (10.18)$$

where e_d is the error due to the disturbance. Since $T(0) = 1$ is required for setpoint tracking purposes, $\lim_{s \rightarrow 0} \frac{T}{C}$ is equal to zero, and thus $e_d(t) \rightarrow 0$ as $t \rightarrow \infty$, only if $C(0) \rightarrow \infty$ as $s \rightarrow 0$. This means that the controller shall have an integral term $\frac{1}{s}$. Note that the introduction of an integrator does not come without drawbacks. Indeed, attention must be paid to the stability of the system since this term introduces a 90-deg phase shift, which tends to reduce

the phase margin. Moreover, windup phenomenon can occur which results in an increase of the error and large transient (if not treated properly) when the actuators saturate. In section [Chapter 10](#) – Control – Review of control methods, these issues will be further discussed and possible solutions to the windup will be proposed.

Noise measurement rejection

One of the drawbacks of the feedback scheme, that has been highlighted, is the injection in the system of the noise due to the measurements, which is detrimental for the performance. Noise n is usually introduced as a high frequency disturbance which affects both the controlled variable y and the control action u .

The relation between y and n is given by the complementary sensitivity function T . It has been already shown that, at low frequency, $\|T\|$ shall be close to 1 for a good setpoint tracking, and this is achieved for high gain of the loop transfer function L . On the other hand, for an effective noise filtering, $\|T\|$ should be as low as possible at high frequency. This condition is met by shaping L in such a way to have low gain at high frequency. Indeed, when $\|L\| \ll 1$, the complementary sensitivity function can be approximated as $T \approx L$.

Measurement noise can be critical also for the control variable because it causes wear and saturation of the actuators due to the induced rapid variations of the action. The response of u to the noise n is captured by the noise sensitivity function CS . Since $T \approx 0$ is required at high frequency, the modulus of sensitivity function S will be close to 1. Therefore, the noise sensitivity function can be approximated as $CS \approx C$. As expected, noise is amplified by high controller gains, and thus imposes a limitation on them. From the formula, it is also clear the importance of filtering possible derivative action in such a way C goes to zero for high frequency.

Loop-shaping design

Stability analysis through Nyquist theorem and margins has been shown to be based on the study of the loop transfer function L . The shape of the sensitivity functions, and thus their characteristics, depends directly on properties of L as well. Hence, it can be concluded that the desired behavior of the closed-loop system can be achieved by properly shaping the open-loop transfer function. This is an important result because, first, it is easier to reason on L than on the closed-loop sensitivity functions, then, the control designer can see immediately the effects of changes in the controller ($L = PC$).

From the analysis of the sensitivity functions, it has been understood that the loop transfer function shall have high gain at low frequency to guarantee good tracking performance and load disturbance attenuation, and low gain at high frequency for limiting the effects of measurement noise. The selection of the slope at low frequency is based on the type of disturbance and reference. It has been explained that L shall have at least one integrator for each integrator of the reference in order to have zero steady-state error. Similarly, an integral action shall be included in the controller to effectively reject a constant disturbance. At high frequency, typically, a slope, or roll-off rate, equal or greater than -2 (i.e., -40 dB/decade on the Bode plot) is desirable. So far, it is not very clear the shape that L should have around the crossover frequency. A fast transition from the high gain at low frequency to the low gain at high frequency could seem attractive. Instead, the slope in the crossover frequency range shall be limited to guarantee adequate robustness properties. As suggested in Ref. [1], the reason can be easily understood considering a minimum phase system, i.e., a system without time delay or poles and zeros in the R.H.P. For these systems, the phase curve can be derived directly from the gain curve of the Bode plot. In particular, in the range in which the slope n is constant, the phase curve has constant value $\sim n\frac{\pi}{2}$. Therefore, the phase margin for these systems can be computed as:

$$\phi_m \approx 180^\circ + 90^\circ n_{gc}, \quad (10.19)$$

where n_{gc} is the constant slope around the crossover frequency. From this relation, it is evident that a slope greater than -1.67 is necessary to have at least a phase margin of 30 deg. Typically, L is shaped in such a way to have a slope of -1 around the crossover frequency. At this point, the typical shape of the open-loop transfer function is well-defined and reported in Fig. 10.9.

Loop-shaping design is an iterative procedure. The open-loop transfer function is shaped in order to meet all the requirements of the application, selecting the gain crossover frequency and adding poles and zeros. The properties of the closed-loop systems are analyzed in terms of stability margins, resonant and sensitivity peaks, bandwidth, etc. If the result is not satisfactory, the L is reshaped to improve it. Generally, the loop-shaping technique is very suitable for SISO systems or system with one input and multiple outputs. As regards the MIMO systems, sometimes, they can be reduced to different SISO systems, which, then, can be treated as explained. When this procedure cannot be performed, other design strategies could be

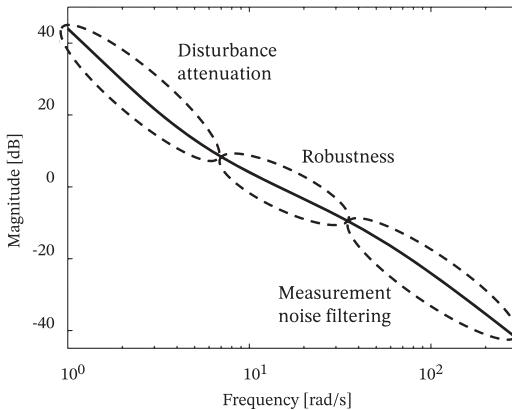


Figure 10.9 Typical shape of the open-loop transfer function $L(s)$.

more suitable. Some of the presented concepts in the frequency domain can be extended to MIMO systems [2]. Alternatively, the design of the controller can be developed in the time domain using the state space representation. This latter approach will be presented in section [Chapter 10 – Control – Control design in state space](#).

To conclude this paragraph on the loop-shaping design, a simple example is provided. Consider the single axis attitude control problem. The dynamics of the system is:

$$\mathbf{I}\ddot{\theta} = \mathbf{t}_c + \mathbf{t}_d, \quad (10.20)$$

where \mathbf{I} is the inertia, θ is the controlled angle, \mathbf{t}_c is the commanded torque, and \mathbf{t}_d is a disturbance torque. Moving in the frequency domain through the Laplace transform, the transfer function of the system is:

$$P = \frac{1}{\mathbf{I}s^2} \quad (10.21)$$

As a first attempt, a simple proportional controller is considered. Therefore, the open-loop transfer function is $L = \frac{k}{\mathbf{I}s^2}$, where k is the controller gain. Clearly, this solution is not adequate. Indeed, the system is not asymptotically stable. A zero is added to the controller leading to:

$$L = k \frac{(s + z_c)}{\mathbf{I}s^2}. \quad (10.22)$$

Assume that the inertia is equal to 10 kgm^2 and the requirements for the controller impose a crossover frequency of at least 5 rad/s . In order to

provide robustness to the system, the zero is chosen to be one decade before of the crossover frequency, $z_c = 1$. A zero in the L.H.P. smaller than the crossover frequency provides an increase of the phase which is beneficial for the phase margin. Afterward, a gain $k = 50$ is set in order to achieve $\omega_{gc} = 5$ rad/s. The gain curve of the open-loop transfer function shows an initial slope equal to -2 due to the double integrator, and then, the slope decreases to -1 around the crossover frequency, thanks to the zero. To improve the rejection to noise, a pole p_c can be added to the controller:

$$L = k \frac{(s + z_c)}{(s + p_c)} \frac{1}{Is^2} \quad (10.23)$$

The pole is selected equal to 25. In this way, the slope of L at high frequency (roll-off rate) becomes -2 , increasing the filtering action of high frequency disturbance. To guarantee $\omega_{gc} = 5$ rad/s, the gain k is increased to 1250. Fig. 10.10 reports the Bode plots of the open-loop transfer function,

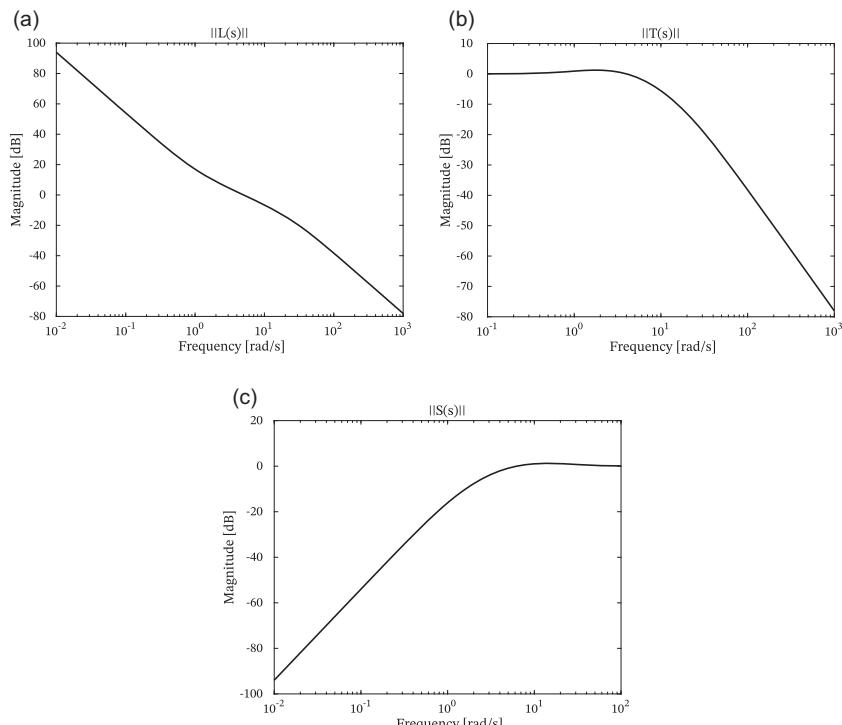


Figure 10.10 Open-loop transfer function (a), complementary sensitivity function (b), and sensitivity function (c) of the single axis attitude example.

the sensitivity function, and the complementary sensitivity function. Basically, the designed controller is a proportional-derivative (PD) controller with a first-order filter. Applying the final value theorem, it can be observed that the error on the controlled variable due to a constant disturbance does not go to zero. To achieve zero steady-state error in this situation, one shall add an integral action and reshape the open-loop transfer function accordingly to guarantee stability and good performance (it is not sufficient to introduce just an integrator $\frac{1}{s}$ in the controller because the system would become unstable). The modification of L to include the integral action is left to the reader.

Feedforward design

Once the feedback has been designed, a feedforward term can be added to improve the response of the system. Indeed, these two strategies are complementary. On one hand, feedback provides robustness to model uncertainties and rejection of unknown disturbance, but it needs the appearance of an error, and thus it acts only after the effects of disturbance or setpoint variation show up. It is a reactive control. On the other hand, feedforward can take corrective actions before disturbances have affected the system, in a predictive way, and it is typically used to obtain a better response to reference signal and to attenuate measurable disturbance. However, it requires a very accurate model. Therefore, one strategy compensates the drawback of the other and vice versa. The combination of feedback and feedforward is called two degrees of freedom control.

In order to show how to design a feedforward, consider the block scheme reported in Fig. 10.11, which includes the feedback loop and two feedforward blocks described by the transfer functions $F_y(s)$ and $F_d(s)$. Both functions provide as output a feedforward term to the control action. The former one receives as input the reference signal, while the latter one receives as input the measurable disturbance. The process transfer function P is split into two transfer functions, $P_1(s)$ and $P_2(s)$, to highlight where the measurable

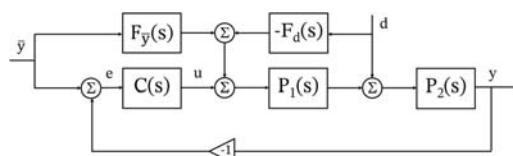


Figure 10.11 Feedback loop with feedforward terms, where $F_y(s)$ and $F_d(s)$ are the reference and disturbance feedforward blocks, respectively.

disturbance enters in the system. The relation between the Laplace transforms of the output and the reference variable is:

$$Y = \frac{PC + PF_{\bar{y}}}{1 + PC} \bar{Y} = [1 + S(PF_{\bar{y}} - 1)] \bar{Y}. \quad (10.24)$$

A perfect tracking means that $Y = \bar{Y}$. To this aim, either the sensitivity function S or the term $(PF_{\bar{y}} - 1)$ should be as small as possible. An ideal tracking would be obtained choosing:

$$F_{\bar{y}} = \frac{1}{P}, \quad (10.25)$$

i.e., using the inverse of the plant model.

A similar procedure can be used to design the feedforward term for attenuating the disturbance effects on the output:

$$Y = \frac{P_2 - F_d P_1 P_2}{1 + PC} D = SP_2(1 - F_d P_1)D. \quad (10.26)$$

Similar to the previous case, a good disturbance rejection is achieved ensuring either the sensitivity function (as already seen in the loop shaping design) or the term $(1 - F_d P_1)$ as small as possible. Ideally, this second condition results in:

$$F_d = \frac{1}{P_1}. \quad (10.27)$$

The procedure to design the feedforward control seems quite straightforward. However, it requires the inversion of the plant model (or part of it), and this process has some important implications to notice. Some issues can arise when the model presents delays, R.H.P. zeros, and a number of poles higher than the zeros. Delays in the inverse mean prediction, which cannot be performed accurately. The inversion of R.H.P. zeros is clearly critical because it would introduce instability in the system. Higher poles than zeros imply differentiation in the inverse, which requires smooth reference and can cause noise amplification. To solve these issues, approximations of the process model shall be used (e.g., focusing only on some frequency range, adding low pass filtering to the inverse, substituting R.H.P. zeros with L.H.P. zeros, etc.). In Ref. [1], these problems related to feedforward design are discussed more thoroughly with examples, and some means of constructing approximate inverses are proposed. Finally, as already pointed out, the knowledge of the model could be poor leading to nonsatisfactory performance. For this reason, it is preferable to use feedforward in combination with feedback, which increases the robustness to uncertainty in the system.

Control design in the state space

So far, the control design in the frequency domain, based on the so-called classical control theory, has been presented. These techniques were particularly popular in the 40s and 50s, and they are still attractive for SISO systems (and MIMO systems, which can be reduced to several SISOs). The analysis of the sensitivity functions provides a deep insight of the characteristics of the closed-loop system, and their features can be directly related to the time behavior. For this reason, performance and robustness specifications are often provided in frequency domain [3,4]. In the 60s and 70s, another important paradigm for the control design emerged, based on the representation of the system dynamics in the time domain by state equations. A key feature of the state-space approach is its natural applicability to MIMO systems. Moreover, it is particularly suitable for optimization. Feedback control law can be obtained directly optimizing a certain performance index. In the following, some time-domain techniques for controller synthesis are presented.

Stability analysis in state space

The stability analysis of an LTI system modeled in state space is straightforward. Consider the generic dynamic system:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}, \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (10.28)$$

where \mathbf{x} is the $n \times 1$ vector of the state, and \mathbf{A} is the $n \times n$ dynamics matrix.

The stability of the system (10.28) is studied looking at the eigenvalues of the dynamics matrix \mathbf{A} , which are the roots of the characteristic polynomial computed as:

$$\lambda = \det(s\mathbf{I}_n - \mathbf{A}). \quad (10.29)$$

The system in Eq. (10.28) is stable if and only if all the eigenvalues of the matrix \mathbf{A} have nonpositive real part, and the geometric multiplicity of any eigenvalue with zero real part is equal to the associated algebraic multiplicity. The system is asymptotically stable if and only if all the eigenvalues have a strictly negative real part. Finally, the system is unstable if any eigenvalue has a strictly positive real part, or a zero real part and the geometric multiplicity are less than the algebraic one. Note that the eigenvalues of the matrix \mathbf{A} correspond exactly to the poles of the closed-loop transfer function in the frequency domain.

State feedback control law

In order to study the state feedback controller, consider the generic state space equation:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu}, \\ \mathbf{y} &= \mathbf{Cx} + \mathbf{Du},\end{aligned}\tag{10.30}$$

where \mathbf{x} is the $n \times 1$ vector of the state, \mathbf{u} is the $m \times 1$ vector of the input, \mathbf{A} is the $n \times n$ dynamics matrix, \mathbf{B} is the $n \times m$ control matrix, \mathbf{C} is the $p \times n$ sensor matrix, and \mathbf{D} is the $p \times m$ direct term. Note that, most of the time, models do not have the direct term, meaning that the output variables are not influenced by the control input. For this reason, in the following, the direct term is set to zero.

Assuming to have all the state available, the state feedback control law takes the following form:

$$\mathbf{u} = -\mathbf{Kx},\tag{10.31}$$

where \mathbf{K} is a $m \times n$ matrix containing the controller gains. Consequently, the closed-loop state equations become:

$$\begin{aligned}\dot{\mathbf{x}} &= (\mathbf{A} - \mathbf{BK})\mathbf{x}, \\ \mathbf{y} &= \mathbf{Cx}.\end{aligned}\tag{10.32}$$

The matrix $(\mathbf{A} - \mathbf{BK})$ represents the new dynamics matrix of the system. The controller in this form, typically called regulator, can be used to shape the transient response for nonzero initial conditions and to attenuate external disturbance to keep the zero-equilibrium state. Later, it will be discussed how to introduce steady-state tracking of a reference.

Pole placement

As already mentioned, the first objective of the controller is to guarantee the stability of the system, and this can be achieved ensuring that all the eigenvalues of $(\mathbf{A} - \mathbf{BK})$ have strictly negative real part. Along with this goal, a designer is required to satisfy some performance in terms of response, such as rise time, overshoot, and settling time. Transient behavior of a dynamic system is closely related to its poles, namely the eigenvalues of the dynamics matrix. Therefore, one way to shape the response in order to obtain the desired transient characteristics is to move the poles of the system by a proper tuning of the gain matrix. This method is called pole placement. It is important to note that this technique requires the availability of the full state and

the controllability of the open-loop state equations (i.e., of the pair (\mathbf{A}, \mathbf{B})) to arbitrarily place the closed-loop eigenvalues.

In the pole placement method, the gains of the controller are selected such that:

$$\det(s\mathbf{I}_n - \mathbf{A} + \mathbf{B}\mathbf{K}) = \lambda^*, \quad (10.33)$$

where λ^* is the desired characteristic polynomial. To illustrate the procedure, consider the following example.

As introduced in [Chapter 4](#) – Orbital Dynamics, the relative motion of a chaser with respect to a target on a circular orbit about the Earth can be expressed in the target local-vertical-local-horizontal (LVLH) frame with the Clohessy–Wiltshire–Hill equations:

$$\begin{aligned} \delta\ddot{x} - 3n^2\delta x - 2n\delta\dot{y} &= u_x, \\ \delta\ddot{y} + 2n\delta\dot{x} &= u_y, \\ \delta\ddot{z} + n^2\delta z &= u_z \end{aligned} \quad (10.34)$$

where n is the orbital rate of the target, and u_x , u_y , u_z are the control actions in the three directions. The LVLH frame is defined with the x -axis pointing out from the Earth to the target spacecraft, the z -axis is normal to the orbital plane, and the y -axis completes the right-handed frame. The motion along the direction orthogonal to the orbital plane is decoupled from the dynamics in the radial and transverse directions. Consequently, two sets of state space equations are considered for the control design:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}_{\mathbf{x}}\mathbf{x} + \mathbf{B}_{\mathbf{x}}\mathbf{u} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 3n^2 & 0 & 0 & 2n \\ 0 & 0 & 0 & 1 \\ 0 & -2n & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u}, \\ \dot{\mathbf{z}} &= \mathbf{A}_{\mathbf{z}}\mathbf{z} + \mathbf{B}_{\mathbf{z}}u_z = \begin{bmatrix} 0 & 1 \\ -n^2 & 0 \end{bmatrix} \mathbf{z} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_z, \end{aligned} \quad (10.35)$$

where $\mathbf{x} = \begin{bmatrix} \delta x & \dot{\delta x} & \delta y & \dot{\delta y} \end{bmatrix}^T$, $\mathbf{u} = [u_x \ u_y]^T$, and $\mathbf{z} = [\delta z \ \dot{\delta z}]^T$. Both systems are controllable. First, the control of the out-of-plane motion is addressed, and the following control action is used:

$$u_z = -\mathbf{k}_z \mathbf{z}, \quad (10.36)$$

where $\mathbf{k}_z = [k_{z1} \ k_{z2}]$ is a row vector of gains. The characteristic polynomial of the closed-loop system can be computed as follows:

$$\det(s\mathbf{I} - \mathbf{A}_z + \mathbf{B}_z \mathbf{k}_z) = s^2 + k_{z2}s + n^2 + k_{z1}, \quad (10.37)$$

In order to achieve the desired closed-loop dynamics, the following characteristic polynomial is prescribed:

$$\lambda^* = s^2 + 2\xi\omega_n s + \omega_n^2, \quad (10.38)$$

Representing a second-order system with damping ξ and natural frequency ω_n . It is recalled that these parameters are strictly related to the time performance of the response, as reported in [Table 10.2](#) and shown in [Fig. 10.8](#). Selected ξ and ω_n to have the desired behavior, the gains of the control can be easily derived accordingly:

$$\begin{aligned} k_{z1} &= \omega_n^2 - n^2, \\ k_{z2} &= 2\xi\omega_n. \end{aligned} \quad (10.39)$$

The gains selection for the in-plane dynamics is a bit more complex since it is a MIMO system. The control action takes the following form:

$$\mathbf{u} = -\mathbf{K}\mathbf{x} = -\begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \end{bmatrix} \mathbf{x}, \quad (10.40)$$

where \mathbf{K} is the control gain matrix. The corresponding closed-loop dynamics matrix becomes:

$$\mathbf{A} - \mathbf{B}\mathbf{K} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 3n^2 - k_{11} & -k_{12} & -k_{13} & 2n - k_{14} \\ 0 & 0 & 0 & 1 \\ -k_{21} & -2n - k_{22} & -k_{23} & -k_{24} \end{bmatrix} \quad (10.41)$$

The number of controller gains is higher than are needed to place the poles at the desired location. Therefore, different solutions are possible. One of them is to choose some gains in order to simplify the dynamics

matrix, and then, to set the remaining ones in order to have the required eigenvalues. For instance, the following initial tuning can be performed:

$$k_{13} = k_{21} = 0, \quad k_{22} = -2n, \quad k_{14} = 2n, \quad (10.42)$$

Then, the resulting dynamics matrix is:

$$\mathbf{A} - \mathbf{BK} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 3n^2 - k_{11} & -k_{12} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -k_{23} & -k_{24} \end{bmatrix}, \quad (10.43)$$

and the associated characteristic polynomial can be computed as:

$$\det(s\mathbf{I}_n - \mathbf{A} + \mathbf{BK}) = (s^2 + k_{12}s - 3n^2 + k_{11})(s^2 + k_{24}s + k_{23}) \quad (10.44)$$

At this point, the desired pole placement can be achieved by imposing the following relation:

$$(s^2 + k_{12}s - 3n^2 + k_{11})(s^2 + k_{24}s + k_{23}) = (s^2 + 2\xi_1\omega_{n1}s + \omega_{n1}) \times (s^2 + 2\xi_2\omega_{n2}s + \omega_{n2}) \quad (10.45)$$

where the damping ξ_1 and ξ_2 and the natural frequencies ω_{n1} and ω_{n2} are chosen to satisfy certain time performance in a similar way to what has been done previously. For example, one could set two conjugate poles significantly faster than the other two in such a way to have the closed-loop dynamics dominated by these latter ones, and thus a second-order dominant response. However, it must be pointed out that moving poles very far from the original ones require high control gains, which could cause amplification of measurement noise and saturation/wear of actuators.

When high-order systems are considered, the procedure to place the poles, illustrated in the example, is not very efficient. A way to simplify it is to formulate the state equation in the controllable canonical form. In particular, considering the SISO system, the dynamics should be expressed by the following equations:

$$\begin{aligned} \dot{\mathbf{x}}_{CF} &= \mathbf{A}_{CF}\mathbf{x}_{CF} + \mathbf{b}_{CF}\mathbf{u}, \\ \mathbf{y}_{CF} &= \mathbf{c}_{CF}\mathbf{x}_{CF}, \end{aligned} \quad (10.46)$$

where:

$$\mathbf{A}_{CF} = \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ -a_n & -a_{n-1} & \cdots & -a_2 & -a_1 \end{bmatrix}, \mathbf{b}_{CF} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad (10.47)$$

$$\mathbf{c}_{CF} = [c_1 \ c_2 \ \cdots \ c_{n-1} \ c_n].$$

The elements of the last row of the dynamics matrix \mathbf{A}_{CF} correspond to the coefficients of the characteristic polynomial of the system:

$$\lambda_{CF} = s^n + a_1 s^{n-1} + a_2 s^{n-2} + \cdots + a_{n-1} s + a_n. \quad (10.48)$$

If the full state controller

$$\mathbf{u} = -\mathbf{k}_{CF} \mathbf{x}_{CF} = -[k_{CF,1} \ k_{CF,2} \ \cdots \ k_{CF,n-1} \ k_{CF,n}] \mathbf{x}_{CF} \quad (10.49)$$

is considered, the closed-loop dynamics matrix becomes:

$$\mathbf{A}_{CF} - \mathbf{b}_{CF} \mathbf{k}_{CF} = \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ -(k_{CF,1} + a_n) & -(k_{CF,2} + a_{n-1}) & \cdots & -(k_{CF,n-1} + a_2) & -(k_{CF,n} + a_1) \end{bmatrix}, \quad (10.50)$$

And the associated characteristic polynomial is:

$$\lambda_{CF} = s^n + (k_{CF,n} + a_1)s^{n-1} + (k_{CF,n-1} + a_2)s^{n-2} + \cdots + (k_{CF,2} + a_{n-1})s + (k_{CF,1} + a_n). \quad (10.51)$$

Imagine having selected the locations of the poles, the desired characteristic polynomial can be written as follows:

$$\lambda^* = s^n + \alpha_1 s^{n-1} + \alpha_2 s^{n-2} + \cdots + \alpha_{n-1} s + \alpha_n. \quad (10.52)$$

The controller gains that transform Eq. (10.51) into Eq. (10.52) can be computed by equating the polynomial:

$$\left\{ \begin{array}{l} k_{CF,1} = \alpha_n - a_n \\ k_{CF,2} = \alpha_{n-1} - a_{n-1} \\ \vdots \\ k_{CF,n-1} = \alpha_2 - a_2 \\ k_{CF,n} = \alpha_1 - a_1 \end{array} \right. \quad (10.53)$$

The procedure with the controllable canonical form is quite straightforward. However, the systems are rarely in this form. Generally, the original state-space formulation is transformed to the canonical form through a linear, nonsingular matrix \mathbf{T} :

$$\mathbf{x}_{CF} = \mathbf{T}\mathbf{x}. \quad (10.54)$$

The dynamics matrix of the original system, \mathbf{A} , \mathbf{b} , and \mathbf{c} are consequently transformed as follows:

$$\begin{aligned} \mathbf{A}_{CF} &= \mathbf{T}\mathbf{A}\mathbf{T}^{-1}, & \mathbf{b}_{CF} &= \mathbf{T}\mathbf{b}, \\ \mathbf{c}_{CF} &= \mathbf{c}\mathbf{T}^{-1}. \end{aligned} \quad (10.55)$$

Afterward, the controller can be designed following the procedure previously explained, and the control action for the original state-space formulation can be determined as:

$$\mathbf{u} = -\mathbf{k}_{CF}\mathbf{x}_{CF} = -\mathbf{k}_{CF}\mathbf{T}\mathbf{x}. \quad (10.56)$$

Considering the relations in Eq. (10.55), the relation between the controllability matrix \mathbf{W} of the original system and the controllability matrix \mathbf{W}_{CF} of the transformed one can be derived, and used to compute the transformation matrix:

$$\begin{aligned} \mathbf{W}_{CF} &= \mathbf{T}\mathbf{W} \\ \mathbf{T} &= \mathbf{W}_{CF}\mathbf{W}^{-1}, \end{aligned} \quad (10.57)$$

where:

$$\mathbf{W}_{\text{CF}} = \begin{bmatrix} a_{n-1} & a_{n-2} & \cdots & a_1 & 1 \\ a_{n-2} & a_{n-3} & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_1 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}^{-1}. \quad (10.58)$$

Hence, the control gain vector \mathbf{k} for the original state-space formulation is expressed as:

$$\mathbf{k} = \mathbf{k}_{\text{CF}} \mathbf{W}_{\text{CF}} \mathbf{W}^{-1}, \quad (10.59)$$

which is called Bass—Gura formula. A multiple-input version of the controllable canonical form is also possible, even if the procedure is much more complicated and is beyond the scope of the book. Once accomplished, the derivation of the state feedback control gains is similar to the one presented. The interested reader can refer to Refs. [5,6].

Thanks to the pole placement method, it is possible to move the poles of a controllable system to any location shaping the closed-loop response as desired. However, there are some issues in this technique. As seen in the example on the relative translational dynamics, in a MIMO system, the designer has to select more control gains than are needed to place the poles. From one hand, these additional parameters provide an increased flexibility in the design. Indeed, not only the poles can be moved to the desired location but also other requirements can be satisfied at the same time. For instance, in the example, they have been chosen to simplify the control structure. In the MATLAB function *place*, they are used to minimize the sensitivity of the closed-loop poles to perturbations in \mathbf{A} or \mathbf{B} . On the other hand, this flexibility results in an indeterminate control solution due to the fact that the desired poles can be achieved with different gain sets, and thus it can be difficult to understand which is the best way to choose them. Moreover, the desirable location for the closed-loop poles may not be known exactly. Finally, with the pole placement method, it is not easy to balance the performance of the system with the control effort required to achieve it. These disadvantages can be overcome by using an alternative technique for the selection of the gains based on optimizing a cost function:

the LQR. This control strategy will be discussed in section [Chapter 10 – Control – Review of control methods](#).

Before proceeding further, a final observation is made. It has been explained that for pole placement, the controllability of the system is required. What happens if the pair (\mathbf{A}, \mathbf{B}) is not controllable? It can be shown that, through a proper change of variables, it is possible to separate the part of the system that is controllable from the one that it is not. The eigenvalues of the former one can be moved designing a stabilizing controller as explained, while the uncontrollable eigenvalues will remain as closed-loop eigenvalues. Consequently, if the uncontrollable eigenvalues have negative real parts, the closed-loop system will be stable, and the pair (\mathbf{A}, \mathbf{B}) (or the open-loop system) is called stabilizable. On the other hand, if the uncontrollable eigenvalues have positive real parts, the closed-loop system will be unstable.

Pole placement for first-, second-, and high-order systems

Linear first- and second-order systems play an important role in control engineering for different reasons. First, simple relations can be found between the position of the poles and the transient response of the system to a step. This information is very useful for the selection of the control gains. Then, first- and second-order systems can be used to approximate high-order ones to some extent.

For first-order system, the transient behavior is described by a single eigenvalue. Considering a stable system, the step response is governed by a decaying exponential with time constant τ , which is the inverse of the eigenvalue (see [Fig. 10.12](#)). The smaller the time constant, the faster the response (i.e., the smaller the rise time). Also, the settling time is related to the eigenvalue, and, especially, it can be estimated as $\sim 4.6\tau$. Therefore, the desired response can be shaped just modifying this single eigenvalue.

On the other hand, the second-order dynamics is characterized by a pair of eigenvalues. In general, they can be expressed as $-\xi\omega_n \pm \omega_n\sqrt{\xi^2 - 1}$, with ξ and ω_n being the damping and the natural frequency. The first parameter determines the shape of the response, the other one the speed (i.e., the larger ω_n , the faster the response). Depending on the damping, the system can be:

- *Overdamped.* $\xi > 1$, real and distinct eigenvalues, slowest transient response.

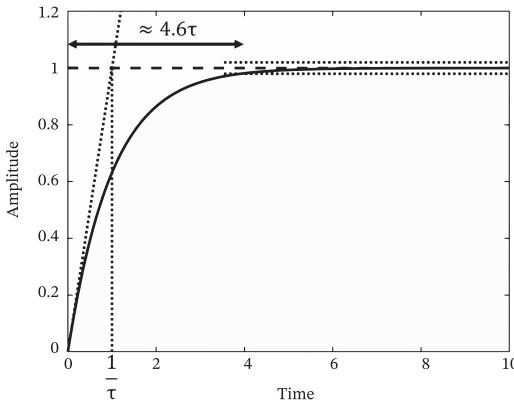


Figure 10.12 First-order response to a step.

- *Critically damped.* $\xi = 1$, real and equal eigenvalues, fastest transient response without overshoot.
- *Underdamped.* $0 < \xi < 1$, complex conjugate eigenvalues, faster transient response (than previous cases) with oscillation and overshoot.
- *Undamped.* $\xi = 0$, complex conjugate eigenvalues on imaginary axis, response with undamped oscillation.
- *Unstable.* $\xi < 0$, at least one eigenvalue with positive real part, unstable response.

The relations between the time performance and the parameters ξ and ω_n have been already introduced, and they are reported in [Table 10.2](#) and illustrated in [Fig. 10.8](#). As shown in the example of the previous paragraph, the control gains can be tuned to have the desired damping and natural frequency, and consequently the desired transient behavior.

The other reason why first- and second-order systems are important for the control design lies in the fact that high-order systems can be approximated by them. Indeed, high-order response are often characterized by some dominant eigenvalues, which are the ones closest to the imaginary axis or, in case of multiple eigenvalues at the same distance, with the lowest damping ratio. Moreover, high-order response can also be shaped to behave as a first- or second-order system. In this case, the designer has to specify the dominant eigenvalues, to meet the requirements, and the location of the remaining ones. As a rule of thumb, these latter one should be 10 times further from the imaginary axis than the dominant eigenvalues. However, this strategy should be applied carefully. Indeed, the further the closed-loop poles from the original ones, the higher the control gains, and

consequently the higher the risk of saturation or wear of actuators. A trade-off must be sought.

Feedforward term

In the previous discussion, it has been shown that the transient behavior of the system can be shaped with a proper tuning of the state-space controller matrix \mathbf{K} (e.g., via pole placement or LQR) in order to achieve the desired time performance. Nevertheless, it has not been discussed yet how to guarantee a good steady-state tracking, which is typically a required feature of the controller. Indeed, the controller in the form of Eq. (10.31) enables to modify the dynamics of the system, but the steady-state value is not directly controlled. To this aim, a feedforward term can be added to the control input, which becomes:

$$\mathbf{u} = -\mathbf{K}\mathbf{x} + \mathbf{K}_{ff}\bar{\mathbf{y}}, \quad (10.60)$$

where \mathbf{K}_{ff} is the $m \times p$ feedforward gain matrix, and $\bar{\mathbf{y}}$ is the $p \times 1$ constant reference vector. Consequently, the closed-loop system can be expressed as:

$$\begin{aligned} \dot{\mathbf{x}} &= (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x} + \mathbf{B}\mathbf{K}_{ff}\bar{\mathbf{y}}, \\ \mathbf{y} &= \mathbf{C}\mathbf{x}, \end{aligned} \quad (10.61)$$

where it can be immediately observed that the additional term does not affect the position of the poles of the system, which is determined only by $(\mathbf{A} - \mathbf{B}\mathbf{K})$. Therefore, feedback and feedforward can be designed independently. The gain matrix \mathbf{K} is selected to guarantee the stability and a certain transient response, while \mathbf{K}_{ff} is chosen to have the desired steady-state value. In particular, at equilibrium, Eq. (10.61) becomes:

$$\begin{aligned} \mathbf{0} &= (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}_e + \mathbf{B}\mathbf{K}_{ff}\bar{\mathbf{y}}, \\ \mathbf{y}_e &= \mathbf{C}\mathbf{x}_e, \end{aligned} \quad (10.62)$$

and, consequently,

$$\mathbf{y}_e = -\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{K})^{-1}\mathbf{B}\mathbf{K}_{ff}\bar{\mathbf{y}}. \quad (10.63)$$

Note that if the control gain \mathbf{K} is designed in order to have an asymptotically stable closed-loop system, the nonsingularity of the matrix $(\mathbf{A} - \mathbf{B}\mathbf{K})$ is guaranteed. At this point, imposing that $\mathbf{y}_e = \bar{\mathbf{y}}$, the following relation is obtained:

$$-\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{K})^{-1}\mathbf{B}\mathbf{K}_{ff} = \mathbf{I}, \quad (10.64)$$

and thus, assuming $p = m$ (i.e., the number of input equals to the number of output), the feedforward gain matrix can be computed as:

$$\mathbf{K}_{ff} = -[\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{K})^{-1}\mathbf{B}]^{-1}. \quad (10.65)$$

If the number of inputs is higher than the output ($m > p$), the Moore–Penrose pseudoinverse is used instead of the inverse. In conclusion, a control law as in Eq. (10.60) with a proper selection of the matrices \mathbf{K} and \mathbf{K}_{ff} enables to modify the dynamics of the system satisfying both steady-state and transient performance objectives.

Integral action

Feedforward term has been introduced to obtain the desired steady-state behavior. However, this method requires accurate knowledge of the system model for a good calibration of \mathbf{K}_{ff} . Uncertainties, parameter variations, or approximation may cause nonnegligible deviation from the target performance. An alternative approach to design a more robust controller with steady-state tracking is to add an integral action to the control input. The control system uses an integrator to provide zero steady-state error, and very accurate model is not necessary anymore. The integral action is introduced augmenting the nominal state equations to include the state \mathbf{x}_I , whose time derivative is equal to:

$$\dot{\mathbf{x}}_I = \mathbf{y} - \bar{\mathbf{y}}, \quad (10.66)$$

In this way, the augmented state equations become:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{x}}_I \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_I \end{bmatrix} + \begin{bmatrix} \mathbf{B} \\ \mathbf{0} \end{bmatrix} \mathbf{u} - \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \bar{\mathbf{y}},$$

$$\mathbf{y} = \mathbf{Cx}. \quad (10.67)$$

Given the augmented system, the state-space control law can be written as:

$$\mathbf{u} = -\mathbf{Kx} - \mathbf{K}_I \mathbf{x}_I, \quad (10.68)$$

and, consequently, the closed-loop dynamics is:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{x}}_I \end{bmatrix} = \begin{bmatrix} \mathbf{A} - \mathbf{BK} & -\mathbf{BK}_I \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_I \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \bar{\mathbf{y}}.$$

$$\mathbf{y} = \mathbf{Cx} \quad (10.69)$$

At this point, the control problem for the augmented system shall be addressed, and the gains \mathbf{K} and \mathbf{K}_I shall be selected in such a way to have an asymptotically stable response. In this way, at equilibrium, $\dot{\mathbf{x}}_I = 0$, and thus the output is equal to the reference realizing zero steady-state error. As a last remark, note that integral action and feedforward term can be combined. The former one guarantees constant disturbance rejection and zero steady-state error even in case of uncertainties in model parameters. The latter one generally improves the transient response to reference signal.

Limitations to control performance

When a control is designed, it is important to be aware of the intrinsic limitations that can be imposed by some properties of the system. This topic has been thoroughly addressed in Refs. [1,7]. In this paragraph, some of the main results are summarized. In particular, first, the Bode's integral formula is reviewed. It shows, through an analysis of the sensitivity function, that certain performance of the closed-loop system cannot be improved over a wide range of frequency. Afterward, the limits due to delays, and zeros and poles in the R.H.P. are discussed.

Bode's integral formula

The sensitivity function provides information about the performance and robustness of the system. In particular, it quantifies the capability of the system of attenuating disturbances. It is recalled that good disturbance rejection is achieved with low gain of the sensitivity function. Moreover, the peak of the sensitivity function is related to the modulus margin, and thus it is a measure of the robustness of the feedback system. A control designer is interested in decreasing the sensitivity function gain, including the peak, as much as possible over a wide range of frequency. However, Bode proved that there are limitations in achieving this goal. Indeed, for a system internally stable and a L such that $\lim_{s \rightarrow \infty} sL = 0$, the following relation holds:

$$\int_0^\infty \log \left\| S(j\omega) \right\| d\omega = \int_0^\infty \log \frac{1}{\|1 + L(j\omega)\|} d\omega = \pi \sum p_k, \quad (10.70)$$

where p_k are poles in the R.H.P. The relation in Eq. (10.70) means that the area under the curve $\log \|S(j\omega)\|$ is constant. Consequently, if the gain of the sensitivity function is decreased for some frequencies to improve disturbance attenuation, it increases, limiting or canceling the effect of the

controller, for other frequencies. This effect is called waterbed effect. Therefore, if a control designer wants a very small sensitivity function at certain frequency, the cost of having an amplification of the disturbance over other frequency has to be paid. The situation is even worse for an unstable system, which has a sensitivity larger than a stable one.

Bode's integral formula confirms the nature of the control design, already highlighted previously. The control designer must look for a balance between conflicting objectives and different performance over the frequency range.

Nonminimum phase systems

Nonminimum phase systems are characterized by the presence of delays, and R.H.P. poles and zeros. These components impose severe constraints on the selection of the gain crossover frequency. For this reason, the control design of nonminimum phase systems is more complex. To understand the restrictions due to nonminimum phase components, some illustrative examples are presented.

Consider a nonminimum phase transfer function P which can be factored as:

$$P = P_{mp}P_{np}, \quad (10.71)$$

where P_{mp} has all the poles and zeros in the L.H.P., while P_{np} has all the nonminimum phase features, and it is chosen to have $\|P_{np}(j\omega)\| = 1$ and negative phase. Assume to have a stabilizing controller C without zeros and poles in the R.H.P. Note that the gain curve of the loop transfer function $L = PC$ corresponds to the gain curve of the minimum phase function $P_{mp}C$ because P_{np} has unit gain over all the frequency. Now consider the case of the presence of an R.H.P. zero represented by $P_{np} = \frac{-s+z}{s+z}$, with $z > 0$. Recalling that for a minimum phase transfer function the phase curve can be derived directly from the gain curve as $n\frac{\pi}{2}$, if a phase margin ϕ_m is required, the following inequality must be satisfied:

$$-\arg P_{np}(j\omega_{gc}) = 2 \arctan\left(\frac{\omega_{gc}}{z}\right) \leq \pi - \phi_m + n_{gc}\frac{\pi}{2}, \quad (10.72)$$

where ω_{gc} is the gain crossover frequency and n_{gc} is the gain crossover slope. From Eq. (10.72), it can be seen that a zero in the R.H.P. limits the achievable value of the gain crossover frequency which shall satisfy:

$$\omega_{gc} \leq z \tan \left(\frac{\pi - \phi_m + n_{gc} \frac{\pi}{2}}{2} \right). \quad (10.73)$$

Note also that slow zeros cause lower gain crossover frequency than fast zeros.

Time delays impose a similar limitation to the gain crossover frequency. Consider $P_{np} = e^{s\tau}$, which represents a pure delay τ in frequency domain. In this case, the inequality to be satisfied is:

$$\omega_{gc}\tau \leq \pi - \phi_m + n_{gc} \frac{\pi}{2}. \quad (10.74)$$

Consequently, the maximum achievable gain crossover frequency is limited as follows:

$$\omega_{gc} \leq \frac{\pi - \phi_m + n_{gc} \frac{\pi}{2}}{\tau}. \quad (10.75)$$

The larger the delay, the lower the gain crossover frequency.

On the other hand, controlling an unstable system, i.e., with poles in the R.H.P., requires larger bandwidth, and thus higher gain crossover frequency. Assume to have $P_{np} = \frac{s+p}{s-p}$, with p being a positive value. If a margin ϕ_m is specified for the application, the phase of the nonminimum transfer function shall be:

$$-\arg P_{np}(j\omega_{gc}) = 2 \arctan \left(\frac{p}{\omega_{gc}} \right) \leq \pi - \phi_m + n_{gc} \frac{\pi}{2}, \quad (10.76)$$

and thus

$$\omega_{gc} \geq \frac{p}{\tan \left(\frac{\pi - \phi_m + n_{gc} \frac{\pi}{2}}{2} \right)}. \quad (10.77)$$

Eq. (10.77) shows that, in order to stabilize and control robustly an unstable system, a high gain crossover frequency is necessary, especially in the case of fast poles. As expected, the control of an unstable system is

challenging and requires not only a large bandwidth of the control, but also of actuators and sensors.

These simple examples should have made clear the difficulties of controlling nonminimum phase systems, due to the restrictions imposed by delays, and R.H.P. poles and zeros. Delay and R.H.P. zeros limit the responsiveness of the closed-loop system. Conversely, the unstable poles require large bandwidth, and thus a very responsive system. Note that the poles are strictly related to the dynamics of the system, and thus the only way to change them is a feedback loop or a redesign of the system. On the other hand, the zeros depend on the how sensors and actuators are integrated in the system. Therefore, adding or moving these elements can modify the position of the zeros, simplifying the control design. Delays are usually introduced in the communication and computation processes. If high control performance is required, the computing and communication systems shall be designed in such a way to minimize the delays.

An introduction to control design for nonlinear systems

The methods and techniques presented so far have been developed for linear systems. However, most of the real applications imply nonlinear dynamics. A question could naturally arise: Can these tools be extended for the control design of a nonlinear system? The answer is yes. The idea is to approximate or convert the nonlinear dynamics to a linear one in such a way to use the previously explained techniques. Following this path, the linearization, gain scheduling, and feedback linearization techniques will be presented.

An alternative approach is the design of the control directly considering the nonlinear system. As for the linear counterpart, it is important to have methods to study the stability of the closed-loop dynamics. To this aim, the Lyapunov stability theorem involving the so-called Lyapunov functions will be introduced. Lyapunov theorem is a powerful tool, which can be used not only to study the stability but also to design a stabilizing controller.

This section provides only few tools and concepts to deal with nonlinear systems, and it is far from covering exhaustively the topic. Some other control methods will be reviewed in the next chapter. However, the reader is invited to deepen the subject through specialized books such as Refs. [8,9].

Linearization

In control engineering, very often a nonlinear system is approximated by a linearized model around an equilibrium condition of interest. The local behavior is studied, the controller is designed to operate and keep the system

in the region of validity of linearization, and the performance is then verified by simulating the closed-loop nonlinear model. For example, in a three-axis stabilized spacecraft, the nonlinear attitude dynamics can be linearized around the nominal orientation and a controller can be designed to keep it, allowing only small variation around the equilibrium point. To illustrate the procedure, consider the following general nonlinear system in state space:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}), \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}, \mathbf{u}),\end{aligned}\quad (10.78)$$

where $\mathbf{f}()$ is a $n \times 1$ nonlinear vector function describing the dynamics, and $\mathbf{h}()$ is a $p \times 1$ nonlinear vector function expressing the relation between the state and output vectors. Assume that an equilibrium solution of the nonlinear system in Eq. (10.78) is given by \mathbf{x}_e and \mathbf{u}_e . The linearized model around the equilibrium point is:

$$\begin{aligned}\delta\dot{\mathbf{x}} &= \mathbf{A}\delta\mathbf{x} + \mathbf{B}\delta\mathbf{u}, \\ \delta\mathbf{y} &= \mathbf{C}\delta\mathbf{x} + \mathbf{D}\delta\mathbf{u},\end{aligned}\quad (10.79)$$

where the state-space matrices are the Jacobians of the nonlinear functions $\mathbf{f}()$ and $\mathbf{h}()$ evaluated in the equilibrium point:

$$\begin{aligned}\mathbf{A} &= \left. \frac{d\mathbf{f}}{d\mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}_e, \mathbf{u}=\mathbf{u}_e} & \mathbf{B} &= \left. \frac{d\mathbf{f}}{d\mathbf{u}} \right|_{\mathbf{x}=\mathbf{x}_e, \mathbf{u}=\mathbf{u}_e} \\ \mathbf{C} &= \left. \frac{d\mathbf{h}}{d\mathbf{x}} \right|_{\mathbf{x}=\mathbf{x}_e, \mathbf{u}=\mathbf{u}_e} & \mathbf{D} &= \left. \frac{d\mathbf{h}}{d\mathbf{u}} \right|_{\mathbf{x}=\mathbf{x}_e, \mathbf{u}=\mathbf{u}_e}\end{aligned}\quad (10.80)$$

and where $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}_e$, $\delta\mathbf{u} = \mathbf{u} - \mathbf{u}_e$, and $\delta\mathbf{y} = \mathbf{y} - \mathbf{h}(\mathbf{x}_e, \mathbf{u}_e)$. Linear state-space equations in Eq. (10.79) approximate the nonlinear dynamics in Eq. (10.78) locally, i.e., around the equilibrium solution. An important theorem, called Lyapunov's indirect method [9], guarantees that if the linearized system is asymptotically stable, i.e., all the eigenvalues of the matrix \mathbf{A} have negative real part, the equilibrium point \mathbf{x}_e is locally asymptotically stable for the nonlinear system. On the other hand, if the linearized system is unstable, i.e., any eigenvalue of \mathbf{A} has positive real part, the equilibrium point \mathbf{x}_e is unstable. Nothing can be said if any eigenvalue is on the imaginary axis. The controller for the nonlinear dynamics can be now designed using the techniques for linear system presented in the previous sections.

Gain scheduling

Linearization is effective if the system operates close to the equilibrium point. Indeed, the stability is guaranteed only locally, and, in any case, the performance can deteriorate significantly for large variations. A technique that has been proposed to overcome this limitation is the gain scheduling. The idea is to linearize the nonlinear model around different equilibrium points, or operating points, to design a controller based on linear control theory for each point, and, finally, to merge the developed controllers in a single one whose parameters change depending on the values of some selected quantities called scheduling variables (e.g., reference or measured variables).

As suggested in Refs. [8,10], the design procedure for gain scheduling can be split into four main steps:

1. Derive a linear parameter-varying model from the nonlinear one. This can be achieved through the classical Jacobian linearization about a family of equilibrium points parametrized by the scheduling variables. An alternative approach is the quasi-linear parameter-varying (quasi-LPV) scheduling, in which the nonlinearities are represented as time-varying parameters, used as scheduling variables.
2. Design the controllers for the linear parameter-varying model using linear control methods. A family of controllers for the plant can be derived directly or it can be the result of an interpolation of the controllers designed at isolated values of the scheduling variables.
3. Implement the family of controllers in such a way that the gains changes with the current values of the selected scheduling variables, and for fixed value of the parameters, some desirable performance are achieved.
4. Assess the performance. Local stability and system properties could be investigated analytically (e.g., studying the linearized system at each equilibrium points). On the other hand, the assessment of nonlocal performance requires the simulation of the nonlinear closed-loop model.

In this paragraph, only the main concepts behind the gain scheduling are summarized. The reader who wants to deepen the knowledge of this technique can refer to Refs. [8,10], where a full description of gain scheduling along with the mathematics can be found.

Feedback linearization

An alternative approach to transform the nonlinear dynamics into an equivalent linear one is the so-called feedback linearization. The idea is to use an inner control loop which linearizes the nonlinear system and introduces a transformed input, whose relationship with the output is linear. Afterward,

techniques for linear system can be exploited to design the outer loop, which completes the control system. Sometimes a transformation of the state variables is necessary along with the change of the input in order to linearize the system. The general theory of feedback linearization is out of the scope of this book, and the interested reader can refer to Refs. [8,9] for a thorough explanation. As an illustrative example, only the application of feedback linearization to the class of mechanical systems of the form:

$$\mathbf{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} + \mathbf{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}}) = \mathbf{B}(\boldsymbol{\chi})\mathbf{u} \quad (10.81)$$

is proposed here since it is relatively frequent, and it is considered relevant in the context of spacecraft control. The $n \times 1$ vector $\boldsymbol{\chi}$ is the configuration of the mechanical system, $\mathbf{M}(\boldsymbol{\chi})$ is the $n \times n$ configuration-dependent inertia matrix, the $n \times 1$ vector $\mathbf{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})$ includes Coriolis, centrifugal, and other nonlinear terms, $\mathbf{B}(\boldsymbol{\chi})$ is the $n \times m$ input matrix, and the $m \times 1$ vector \mathbf{u} is the usual control action. Assuming $n = m$, the nonlinear model in Eq. (10.81) can be transformed into the linear one choosing:

$$\mathbf{u} = \mathbf{B}(\boldsymbol{\chi})^{-1} (\mathbf{M}(\boldsymbol{\chi})\mathbf{v} + \mathbf{C}(\boldsymbol{\chi}, \dot{\boldsymbol{\chi}})), \quad (10.82)$$

where \mathbf{v} is the transformed input. The resulting closed-loop system becomes:

$$\mathbf{M}(\boldsymbol{\chi})\ddot{\boldsymbol{\chi}} = \mathbf{M}(\boldsymbol{\chi})\mathbf{v}, \quad (10.83)$$

and, consequently,

$$\ddot{\boldsymbol{\chi}} = \mathbf{v}, \quad (10.84)$$

which is a linear system. Linear control theory can be now used to design the control law for outer loop with input \mathbf{v} . Note that, in contrast to linearization, no approximations have been introduced. However, an exact cancellation of nonlinearities is achieved only with perfect knowledge of the system model, which is almost impossible to have. Typically, uncertainties introduce perturbations which deteriorate the control performance with respect to the ideal case. If the level of uncertainties is high, adaptive control scheme can be integrated to feedback linearization to improve system response. As a final remark, feedback linearization should be used carefully. First, generally, not all the nonlinear systems can be linearized via feedback, as explained in Refs. [8,9]. Then, even if a system is linearizable, some nonlinear terms may be beneficial, and thus canceling them out should be avoided.

Stability analysis for nonlinear systems

Previous techniques are based on the idea of reducing in some way the nonlinear system to a linear one (or a family of linear ones), and then applying tools for linear system to design the control. A different approach consists in the synthesis of the control considering the nonlinear model directly. For this purpose, an important result to study the stability is the Lyapunov's direct method [9]. This theorem is inspired by a simple physical observation: if the total energy is continuously dissipated, a system must eventually reach an equilibrium point.

To introduce the theorem, consider the system:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}). \quad (10.85)$$

A scalar energy-like function $V(\mathbf{x})$, called Lyapunov function, is defined. This function is characterized by some properties. First, it has to be positive definite, i.e., $V(\mathbf{x}) > 0$ for $\mathbf{x} \neq 0$ and $V(0) = 0$, and have a continuous first derivative. Then, its derivative along trajectories, $\dot{V}(\mathbf{x})$, is negative semidefinite, i.e., $\dot{V}(\mathbf{x}) \leq 0$ for $\mathbf{x} \neq 0$. Lyapunov's direct method states that if there exists a Lyapunov function for the system in Eq. (10.85), and $\dot{V}(\mathbf{x})$ is strictly negative ($\dot{V}(\mathbf{x}) < 0$ for $\mathbf{x} \neq 0$), the equilibrium point at the origin is asymptotically stable. More precisely, if the conditions are satisfied only locally, it is said that the equilibrium is locally asymptotically stable. If the additional condition $V(\mathbf{x}) \rightarrow \infty$ as $\|\mathbf{x}\| \rightarrow \infty$ is met, the asymptotic stability is global. In many cases, it is quite hard to satisfy Lyapunov conditions directly because it is not easy to find a Lyapunov function such that $\dot{V}(\mathbf{x}) < 0$, and only $\dot{V}(\mathbf{x}) \leq 0$ can be shown. However, thanks to another theorem, the LaSalle invariant set theorem [8], it is possible to prove the asymptotic stability also for these situations. In particular, let us assume $\dot{V}(\mathbf{x}) \leq 0$ over the entire state space and $V(\mathbf{x}) \rightarrow \infty$ as $\|\mathbf{x}\| \rightarrow \infty$, and denote by \mathbf{R} the set of points for which $\dot{V}(\mathbf{x}) = 0$ and by \mathbf{M} the largest invariant set in \mathbf{R} . LaSalle theorem states that all solutions globally asymptotically stable converge to \mathbf{M} as $t \rightarrow \infty$. From this theorem, an important, and frequently used, corollary is derived and states that if \mathbf{R} contains no other trajectories than the trivial on $\mathbf{x} = 0$, the equilibrium point in the origin is asymptotically stable.

Lyapunov's direct method is primarily an important tool to study the stability of a nonlinear system. However, it is also very useful to design stabilizing controllers $\mathbf{u}(\mathbf{x})$. There are two main approaches based on trial-and-error procedure. In the first technique, a certain form of the control input is chosen, and then the Lyapunov theorem is used to prove the stability of the

closed-loop system. In the second technique, a Lyapunov function is hypothesized, and then a controller is designed to realize it and make sure $\dot{V}(\mathbf{x}) < 0$ (or $\dot{V}(\mathbf{x}) \leq 0$ with LaSalle conditions). An example of both methods will be proposed in the next section.

Attitude regulation example

Consider the attitude dynamics of a satellite, whose describing equations are nonlinear:

$$\mathbf{I}\ddot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} = \mathbf{t}. \quad (10.86)$$

The 3×3 inertia matrix is denoted by \mathbf{I} , the angular velocity by $\boldsymbol{\omega}$, and the control torques by \mathbf{t} . For the example, a diagonal inertia matrix is assumed, and the goal of the control is to reorient the satellite from an initial attitude to a final one. The first control law that is considered uses Euler angles ϑ_x , ϑ_y , and ϑ_z as attitude parameters, and it is based on the linearization of the dynamics around the equilibrium point $\vartheta_x = \vartheta_y = \vartheta_z = 0$. The linearized model becomes:

$$\begin{aligned} I_x \ddot{\vartheta}_x &= t_x, \\ I_y \ddot{\vartheta}_y &= t_y, \\ I_z \ddot{\vartheta}_z &= t_z, \end{aligned} \quad (10.87)$$

Basically, the initial nonlinear control problem is transformed into three single-axis control problem similar to the example case considered for loop shaping. Therefore, the control law can be designed in an analogous fashion via loop shaping (or, alternatively, via pole placement). In particular, a PD controller as in Eq. (10.22) is considered, and thus the closed-loop system is:

$$\begin{aligned} I_x \ddot{\vartheta}_x &= K_{px}(\bar{\vartheta}_x - \vartheta_x) - K_{vx}\dot{\vartheta}_x, \\ I_y \ddot{\vartheta}_y &= K_{py}(\bar{\vartheta}_y - \vartheta_y) - K_{vy}\dot{\vartheta}_y, \\ I_z \ddot{\vartheta}_z &= K_{pz}(\bar{\vartheta}_z - \vartheta_z) - K_{vz}\dot{\vartheta}_z, \end{aligned} \quad (10.88)$$

where $\bar{\vartheta}_x$, $\bar{\vartheta}_y$, and $\bar{\vartheta}_z$ are the setpoints for the Euler angles, and K_p* and K_v* are the proportional and derivative gains. The control parameters shall be selected to have a stable system and certain desired response. Since a linearization procedure has been followed, the stability and good performance are guaranteed only locally around the equilibrium point. To appreciate this fact, another control law, which can ensure global asymptotic stability, is

introduced for comparison. In this case, the quaternion parameterization is used, and the following controller is assumed:

$$\mathbf{t} = -2\mathbf{H}^T \mathbf{K}_p \delta \mathbf{q}_{1:3} - \mathbf{K}_v \boldsymbol{\omega}, \quad (10.89)$$

leading to the closed-loop dynamics:

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} = -2\mathbf{H}^T \mathbf{K}_p \delta \mathbf{q}_{1:3} - \mathbf{K}_v \boldsymbol{\omega}, \quad (10.90)$$

where $\delta \mathbf{q}_{1:3}$ is the vector part of the quaternion error, \mathbf{K}_p and \mathbf{K}_v are 3×3 diagonal matrices containing positive control gains, and \mathbf{H} is the 3×3 matrix mapping $\boldsymbol{\omega}$ into $\delta \dot{\mathbf{q}}_{1:3}$, namely:

$$\delta \dot{\mathbf{q}}_{1:3} = \frac{1}{2} \mathbf{H} \boldsymbol{\omega} = \frac{1}{2} ([\delta \mathbf{q}_{1:3}] + \delta q_4 \mathbf{I}_3) \boldsymbol{\omega}. \quad (10.91)$$

The variable δq_4 is the scalar part of the quaternion error. Note that the considered quaternion error dynamics assumes that the setpoint is constant. In order to prove the stability of the closed-loop system, the following candidate Lyapunov function is defined:

$$V = \frac{1}{2} \boldsymbol{\omega}^T \mathbf{I} \boldsymbol{\omega} + 2\delta \mathbf{q}_{1:3}^T \mathbf{K}_p \delta \mathbf{q}_{1:3}, \quad (10.92)$$

which is positive definite. The time derivative can be computed as follows:

$$\begin{aligned} \dot{V} &= \boldsymbol{\omega}^T \mathbf{I} \dot{\boldsymbol{\omega}} + 4\delta \mathbf{q}_{1:3}^T \mathbf{K}_p \delta \dot{\mathbf{q}}_{1:3} \\ &= -\boldsymbol{\omega}^T (\boldsymbol{\omega} \times \mathbf{I} \boldsymbol{\omega} + 2\mathbf{H}^T \mathbf{K}_p \delta \mathbf{q}_{1:3} + \mathbf{K}_v \boldsymbol{\omega}) + 2\delta \mathbf{q}_{1:3}^T \mathbf{K}_p \mathbf{H} \boldsymbol{\omega}, \end{aligned} \quad (10.93)$$

where the closed-loop dynamics in Eq. (10.90) and the quaternion error dynamics in Eq. (10.91) have been used in the derivation. Expanding Eq. (10.93), it is obtained:

$$\dot{V} = -\boldsymbol{\omega}^T \mathbf{K}_v \boldsymbol{\omega} \leq 0, \quad (10.94)$$

which means that \dot{V} is a negative semidefinite function. Indeed, $\dot{V} = 0$ if $\boldsymbol{\omega} = 0$, but $\delta \mathbf{q}_{1:3}$ could be potentially anything. However, from the dynamics in Eq. (10.90), it can be observed that the condition $\dot{V} = 0$, and thus $\boldsymbol{\omega} = 0$, can be maintained by the system only if $\delta \mathbf{q}_{1:3} = 0$. Therefore, applying LaSalle theorem, it is possible to conclude that the origin $\boldsymbol{\omega} = 0$, $\delta \mathbf{q}_{1:3} = 0$ is asymptotically stable. The controller in Eq. (10.89) can reorient the spacecraft to a desired orientation starting from any initial orientation. The attitude control examples are further discussed in Chapter 14 – Applicative GNC cases and examples.

The two control laws are compared in simulation considering the following parameters: $I_x = 900 \text{ kg/m}^2$, $I_y = 500 \text{ kg/m}^2$, $I_z = 650 \text{ kg/m}^2$, $K_{px} = 900$, $K_{py} = 500$, $K_{pz} = 650$, $K_{vx} = 1440$, $K_{vy} = 800$, and $K_{vz} = 1040$. An approach similar to the one used in Ref. [11] for the comparison of some basic attitude control law is followed, and two test case scenarios are presented. In the first one, a small attitude maneuver is commanded as a step in the Euler angles ($\bar{\phi} = \bar{\theta}_x = 7 \text{ deg}$, $\bar{\theta} = \bar{\theta}_y = -4 \text{ deg}$, and $\bar{\psi} = \bar{\theta}_z = 2 \text{ deg}$). The results of the simulations are reported in Fig. 10.13. It can be seen that the two controllers provide very similar behavior and reach zero steady-state error. As expected, the linearized model is a good approximation of the nonlinear dynamics close to the equilibrium point. The story is different if a large reorientation of the spacecraft is required. Fig. 10.14 shows the response of the closed-loop system to a larger step: $\bar{\phi} = 70 \text{ deg}$, $\bar{\theta} = -40 \text{ deg}$, and $\bar{\psi} = 20 \text{ deg}$. It is evident the superiority of the quaternion-based controller which guarantees a well-behaved response also in this scenario. This is because the design of the controller took into account the nonlinear nature of the dynamics, and a solution ensuring global asymptotic stability has been found. On the other hand, the controller based on linearization has been developed on an approximation of the dynamics, which is good enough only around the equilibrium point. As soon as a large deviation is requested, the performance deteriorates.

Other control laws based on quaternions can be designed for attitude regulation, which guarantee global asymptotic stability. For instance, in Ref. [12], Lyapunov theorem is used to derive a class of nonlinear PD controllers. A candidate Lyapunov function is defined as:

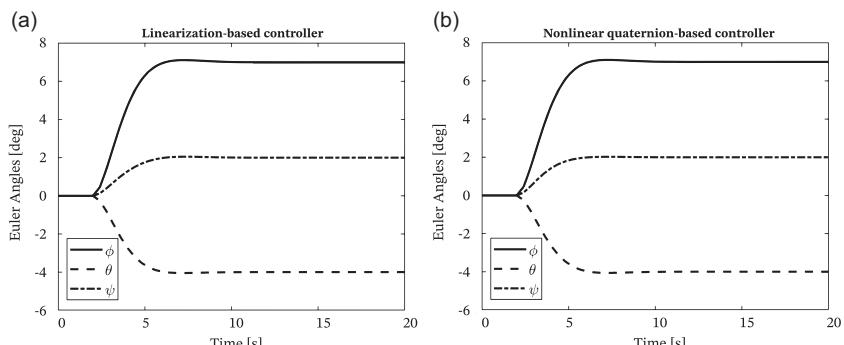


Figure 10.13 Comparison between linearization-based controller with Euler angle error (a) and nonlinear quaternion-based controller (b). Small attitude maneuver.

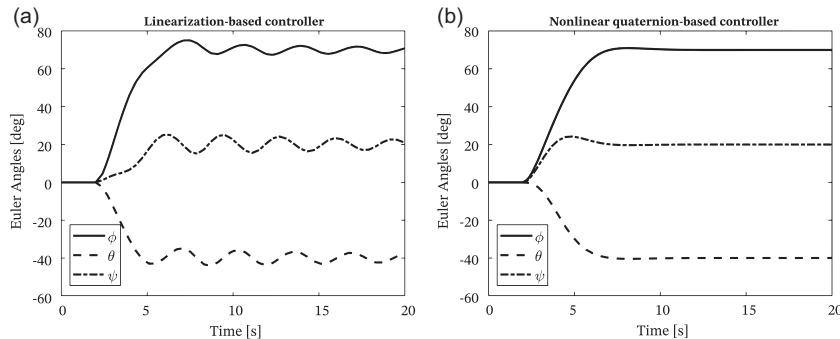


Figure 10.14 Comparison between linearization-based controller with Euler angle error (a) and nonlinear quaternion-based controller (b). Large attitude maneuver.

$$V = \frac{1}{2} \boldsymbol{\omega}^T \mathbf{I} \boldsymbol{\omega} + 2kU(\delta q_4), \quad (10.94)$$

where $k > 0$ is a control parameter and $U(\delta q_4)$ is a nonnegative function which is zero in $\delta q_4 = \pm 1$. In order to guarantee global asymptotic stability, the following control law is derived:

$$\mathbf{t} = 2k \frac{\partial U(\delta q_4)}{\partial \delta q_4} \delta \mathbf{q}_{13} - \mathbf{K}_v \boldsymbol{\omega}. \quad (10.95)$$

Different controller can be developed depending on the selection of the function $U(\delta q_4)$. Note that the asymptotically stable equilibrium points change with $U(\delta q_4)$ as well. For example, if $U(\delta q_4) = 1 - \delta q_4^2$, the controller becomes:

$$\mathbf{t} = -2k\delta \mathbf{q}_{13}\delta q_4 - \mathbf{K}_v \boldsymbol{\omega}, \quad (10.96)$$

and the asymptotically stable equilibrium points are $\delta q_4 = \pm 1$, while the unstable one is $\delta q_4 = 0$. In Ref. [12], other selection of $U(\delta q_4)$ is proposed.

➤ Review of control methods

This section presents a brief review of the most useful control methods for GNC applications. Specifically, PID control, LQR, adaptive controllers, robust controllers, MPC, and SMC are discussed.

PID control

PID are certainly the most widespread controllers both on ground, in industry, and in space, on satellites. Their name recalls the three basic terms making up the control action: the proportional term (P), the integral term (I), and the derivative term (D). Their popularity is due to several factors. They can be used to control a variety of dynamic systems, spanning from low-order to high-order, and from linear to nonlinear. Control parameters are few and relatively simple to set, and automatic tuning methods are available. They do not rely on very accurate mathematical models, which are typically necessary for advanced control strategies and whose development and parameters identification require time and resources. Moreover, note that, sometimes, the control quality of advanced strategies is limited by the performance of sensors, actuators, or communication system, and thus they may not provide significant advantages with respect to PID. Finally, PID are often exploited in hierarchical control architecture as low-level controllers.

The ideal PID control action u takes the following form:

$$u = k_p e + k_i \int_0^t e(\tau) d\tau + k_d \frac{de}{dt} \quad (10.97)$$

where e is the error between the reference \bar{y} and the controlled variable y , k_p is the proportional gain, k_i is the integral gain, and k_d is the derivative gain. Control laws can be designed also considering only some of the three terms in Eq. (10.97). For example, it is possible to have P, I, PI, or PD controllers. Let consider one term at the time to understand what contribution each term provides. The control scheme in Fig. 10.15 is used in support of the discussion.

The proportional term realizes an algebraic relation between the control input and the current control error. As soon as the system deviates from the

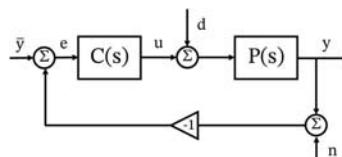


Figure 10.15 General feedback control scheme. $C(s)$ is the controller transfer function, $P(s)$ is the plant transfer function, \bar{y} is the reference signal, y is the controlled variable, u is the control action, d is the load disturbance, and n is the measurement noise.

reference behavior, a corrective action is applied. However, in general, the proportional term is not sufficient to reach zero steady-state error. Indeed, assuming $P(0) = \text{const} \neq 0$, a proportional controller $C(s) = k_p$, a constant reference and disturbance of amplitude $A_{\bar{y}}$ and A_d , respectively, and neglecting the noise, the steady-state error can be computed with the final value theorem and results to be:

$$\lim_{t \rightarrow \infty} e = \lim_{s \rightarrow 0} se = \frac{A_{\bar{y}}}{1 + k_p P(0)} - \frac{P(0) A_d}{1 + k_p P(0)}. \quad (10.98)$$

where s is the generalized frequency introduced in [Chapter 10 – Control – Control design](#) (see also [Appendix – A2](#) for an introduction to Laplace transform). From [Eq. \(10.98\)](#), it is evident that the steady-state error does not go to zero and depends on the amplitude of the input, and the gain of the proportional controller. In particular, the larger the control parameter, the smaller the error. Nevertheless, it is important to bear in mind that, increasing too much k_p leads to noise amplification and can make the system become oscillatory, even unstable.

As discussed in the section [Chapter 10 – Control – Control design](#), a way to guarantee a zero steady-state error, for a step variation of the reference and the disturbance, is the introduction of an integral action. The transfer function of a PI controller is $C(s) = k_p \frac{(1+T_i s)}{T_i s}$, with $T_i = \frac{k_p}{k_i}$, and thus [Eq. \(10.98\)](#) becomes:

$$\lim_{t \rightarrow \infty} e = \lim_{s \rightarrow 0} se = \lim_{s \rightarrow 0} \frac{A_{\bar{y}} T_i s}{T_i s + k_p (1 + T_i s) P(0)} - \frac{P(0) A_d T_i s}{T_i s + k_p (1 + T_i s) P(0)} = 0. \quad (10.99)$$

This result can also be proved in a more general way [1]. Assume that u and e converges at steady state to an equilibrium point $u = u_e$ and $e = e_e$. The control action can be written as:

$$u_e = k_p e_e + k_i \lim_{t \rightarrow \infty} \int_0^t e(\tau) d\tau \quad (10.100)$$

The only way to have the right-hand side finite is that $e(t)$ converges to zero, and consequently $e_e = 0$. Note that no assumptions about linearity or time invariance has been made, and only the existence of an equilibrium point at steady-state has been assumed.

Increasing the integral gain improves reference tracking and disturbance attenuation, but, as before, an excessive increase can lead to oscillations of the response. Notice that there are also other drawbacks in using the integral term. First, it introduces a 90-deg phase shift, which could destabilize the system. Then, it can cause the so-called integrator windup. This phenomenon occurs when an actuator saturates. The feedback loop is broken, and the system evolves in open loop. The error is generally nonzero, and the integral term tends to increase. Consequently, the control action may become very large, and the control signal remains saturated. It may take a long time before the control action and the integral term return to values useful for a proper control, and this causes large transient. In order to avoid this issue, antiwindup strategies shall be implemented. The easiest solution is to “switch off” the integral action when an actuator saturates. To this aim, either specific devices can be used to monitor the status of the actuator, or the output of the controller is saturated (e.g., via software). This latter strategy does not require additional sensors, but the limits shall correspond as much as possible to the limits of the actuator in order to guarantee the antiwindup and, at the same time, to fully exploit the actuation system. Note that windup is related to the use of an integral action regardless of the specific control law (i.e., PID, LQR, etc.) In Ref. [1], an example of an antiwindup method for output feedback controllers can be found along with a different antiwindup implementation for PID.

The last term in a PID control law is the derivative one, which provides a sort of predictive action. To appreciate this, the control action in Eq. (10.97) is rewritten as:

$$u = k_p \left(e + T_d \frac{de}{dt} \right) + \frac{k_p}{T_i} \int_0^t e(\tau) d\tau = k_p e_{pd} + \frac{k_p}{T_i} \int_0^t e(\tau) d\tau \quad (10.101)$$

where $T_d = \frac{k_d}{k_p}$, and the variable $e_{pd} = e + T_d \frac{de}{dt}$ can be interpreted as a prediction of the error at time $t + T_d$. Generally, the derivative action introduces some damping in the system. This can be clearly seen in the spacecraft single-axis control problem, where a simple proportional term is not sufficient for asymptotic stability and a derivative action is necessary (see section Chapter 10 – Control – Control design – Loop-shaping design). When implementing the derivative term, it is important to take into account that measurement noise, usually at high frequency, will be amplified causing

saturation and/or wear of the actuators. Therefore, a low-pass filter is usually associated to this term. The Laplace transform of Eq. (10.97) becomes:

$$u(s) = k_p \left(1 + \frac{1}{sT_i} + \frac{T_d s}{1 + \frac{T_d}{N}s} \right) e(s), \quad (10.102)$$

where N usually varies between 5 and 20.

The parameters of the PID can be tuned in several ways. For LTI systems, both loop-shaping and pole placement techniques, presented in section Chapter 10 – Control – Control design, can be used. For a first-order dynamics, a PI is sufficient to move the closed-loop poles to the desired location, while, in case of a second-order system, PID controller is necessary. PI, PD, or PID can be also used with high-order systems, with the limitation that only dominant poles can be placed. Even nonlinear dynamics can be controlled with PID control laws through linearization techniques or directly (see section Chapter 10 – Control – Control design). For the stability analysis, Lyapunov and LaSalle theorems can be used. The tuning of the gains can be performed also through empirical methods such as Ziegler and Nichols' method [13], or relay feedback, as explained in Ref. [1].

Linear quadratic regulator

In section Chapter 10 – Control – Control design, the pole placement method has been presented to select the gains of a state feedback controller, and some shortcomings of this technique have been highlighted. In particular, in case of MIMO systems, the parameters are more than are needed to move the eigenvalues, and thus it is not always easy to tune the controller, or to understand which is the best solution. Moreover, a balance between the required performance and the magnitude of the control input is often difficult.

In order to overcome these limitations, optimal control has been developed. The idea is formulating the control problem as an optimization problem, which can indeed capture the fundamental trade-off between regulation performance and control effort. The most popular optimal control problem is the LQR, whose main results for LTI systems are reviewed in the following. For a more in-depth study and all the mathematical proofs, the reader is invited to refer to Refs. [2,5], where extension to linear time-varying systems is also covered.

Finite-horizon linear quadratic regulator

Consider the LTI system in the state space:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}, \quad \mathbf{x}(0) = \mathbf{x}_0 \quad (10.103)$$

And the control input in the following form:

$$\mathbf{u} = -\mathbf{Kx}. \quad (10.104)$$

The control gain matrix \mathbf{K} is determined minimizing the following cost function:

$$\mathcal{J} = \int_0^{t_f} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} dt + \mathbf{x}_f^T \mathbf{Z} \mathbf{x}_f, \quad (10.105)$$

where \mathbf{Q} , \mathbf{R} , and \mathbf{Z} are weighting symmetric matrices, with \mathbf{Q} and \mathbf{Z} positive semidefinite and \mathbf{R} positive definite, and $\mathbf{x}_f = \mathbf{x}(t_f)$, with t_f being the final time of the interval of interest. It can be proved that the control input minimizing the cost function in Eq. (10.105) is:

$$\mathbf{u} = -\mathbf{Kx} = -\mathbf{R}^{-1} \mathbf{B}^T \mathbf{S} \mathbf{x}, \quad (10.106)$$

where \mathbf{S} is a positive definite, symmetric matrix given by the differential Riccati equation:

$$-\dot{\mathbf{S}} = \mathbf{SA} + \mathbf{A}^T \mathbf{S} - \mathbf{SBR}^{-1} \mathbf{B}^T \mathbf{S} + \mathbf{Q}, \quad \mathbf{S}(t_f) = \mathbf{Z}, \quad (10.107)$$

which is solved backward in time from the final condition to obtain the matrix \mathbf{S} for the entire time interval. Note that \mathbf{S} varies in time, and thus the final control law is linear time varying.

The design process consists in selecting iteratively the weighting matrices \mathbf{Q} , \mathbf{R} , and \mathbf{Z} , starting from a guess, and simulating the closed-loop response. If the performance is not satisfactory, the weighting matrices are changed, and the process repeated until the desired objectives are achieved. The cost function in Eq. (10.105) describes mathematically the design trade-off between the control effort and the closed-loop performance. If the weighting matrices are selected in such a way to penalize more the state than the control, the closed-loop system will move rapidly to the equilibrium point at the origin, but it will require a high control effort. On the other hand, if the quadratic term involving the input is larger than the one of the state, the control energy will be limited, but the regulation performance will be poor. Typically, diagonal matrices are used as weights. The diagonal elements describe how much each state and control input contribute to the overall

cost. When a particular state or input is wanted to be penalized, the associated diagonal element is chosen larger than the other ones.

Infinite-horizon linear quadratic regulator

If the restriction of having a finite final time t_f is removed, the LQR problem is simplified. Indeed, the cost function becomes:

$$\mathcal{J} = \int_0^{\infty} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{u}^T \mathbf{R} \mathbf{u} dt, \quad (10.108)$$

and the control action takes the same form as in [Eq. \(10.106\)](#), but the matrix \mathbf{S} is constant (for LTI systems) and can be found as solution of the algebraic Riccati equation:

$$\mathbf{SA} + \mathbf{A}^T \mathbf{S} - \mathbf{SBR}^{-1} \mathbf{B}^T \mathbf{S} + \mathbf{Q} = \mathbf{0}. \quad (10.109)$$

It is worth mentioning an interesting property of this control law. It can be proved that if the pair (\mathbf{A}, \mathbf{B}) is stabilizable, and the pair $(\sqrt{\mathbf{Q}}, \mathbf{A})$ is observable, then, there exists only a unique solution \mathbf{S} of [Eq. \(10.109\)](#), and the closed-loop system is asymptotically stable. This is a remarkable property considering the difficulties of designing a stabilizing controller for large MIMO systems. Finally, it is recalled that feedforward input and integral action can be added to both finite- and infinite-horizon LQR control laws for steady-state tracking, as explained in section [Chapter 10 – Control – Control design – Control design in state space](#).

Linear quadratic Gaussian control

When the state feedback control has been introduced, it has been assumed that the full state is available. However, in practice, it can happen that only a subset of the state vector can be measured. In these situations, can state feedback control be implemented? The answer is yes. Indeed, in some cases, the accessible subset can be used to estimate the nonmeasurable one through an observer. Consider the usual dynamics in state space:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu}, \mathbf{y} = \mathbf{Cx}. \quad (10.110)$$

The equation of the linear observer takes the following form:

$$\dot{\tilde{\mathbf{x}}} = \mathbf{A}\tilde{\mathbf{x}} + \mathbf{Bu} + \mathbf{L}(\mathbf{y} - \mathbf{Cx}), \quad (10.111)$$

where $\tilde{\mathbf{x}}$ is the estimated state vector, and \mathbf{L} is the observer gain matrix. From Eqs. (10.110) and (10.111), the dynamics of the observation error $\delta\mathbf{x}$ can be derived as:

$$\dot{\delta\mathbf{x}} = \dot{\mathbf{x}} - \dot{\tilde{\mathbf{x}}} = (\mathbf{A} - \mathbf{LC})\delta\mathbf{x}. \quad (10.112)$$

The error goes to zero if \mathbf{L} is selected in such a way that all the eigenvalues of the matrix $\mathbf{A} - \mathbf{LC}$ have negative real part. Note that \mathbf{L} can be selected using the pole placement method explained in section Chapter 10 – Control – Control design – Control design in state space. Indeed, the observer design problem can be seen as a state feedback control problem (i.e., find \mathbf{K} so that $\mathbf{A} - \mathbf{BK}$ has given eigenvalues) with \mathbf{A}^T instead of \mathbf{A} , \mathbf{C}^T instead of \mathbf{B} , and \mathbf{L}^T instead of \mathbf{K} . Like the control problem, the system must be observable in order to arbitrarily locate the eigenvalues of $\mathbf{A} - \mathbf{LC}$.

The feedback control law is now introduced using the estimate of the state:

$$\mathbf{u} = -\mathbf{K}\tilde{\mathbf{x}}, \quad (10.113)$$

and thus, the dynamics in Eq. (10.111) becomes:

$$\dot{\mathbf{x}} = \mathbf{Ax} - \mathbf{BK}\tilde{\mathbf{x}} = (\mathbf{A} - \mathbf{BK})\mathbf{x} + \mathbf{BK}\delta\mathbf{x}. \quad (10.114)$$

Consequently, the closed-loop system is governed by:

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\delta\mathbf{x}} \end{bmatrix} = \begin{bmatrix} \mathbf{A} - \mathbf{BK} & \mathbf{BK} \\ 0 & \mathbf{A} - \mathbf{LC} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \delta\mathbf{x} \end{bmatrix}, \quad (10.115)$$

and its characteristic polynomial can be computed as:

$$\lambda(s) = \det(s\mathbf{I} - \mathbf{A} + \mathbf{BK})\det(s\mathbf{I} - \mathbf{A} + \mathbf{LC}). \quad (10.116)$$

Eq. (10.116) reveals that the design of the control and the observer can be carried out independently. Indeed, the poles of the controller do not affect the poles of the observer (and vice versa), and, given asymptotically stable controller and observer, their interconnection is asymptotically stable. This result is known as separation principle. Consequently, one is free to design first the observer, and then the control law, which receives as input the estimated state vector as in Eq. (10.113), or vice versa.

Previously, it has been explained that pole placement can be used for the selection of the observer gain matrix \mathbf{L} , but this is not the only method. For a fast convergence of the state reconstruction, observer poles should be located deep in the left-half complex plane. Generally, this means having the gain matrix \mathbf{L} large, which makes the observer sensitive to measurement noise.

A compromise between the speed of reconstruction and the immunity to noise is desirable and should be sought. To this aim, consider the system in Eq. (10.110), to which process noise \mathbf{v} and measurement noise \mathbf{w} have been added:

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} + \mathbf{v}, \quad (10.117)$$

$$\mathbf{y} = \mathbf{Cx} + \mathbf{w}.$$

with $\mathbf{x}_0 = E\{\mathbf{x}(0)\}$. It is assumed that both \mathbf{v} and \mathbf{w} are Gaussian white noise with zero mean and covariance \mathbf{P}_v and \mathbf{P}_w , respectively. Moreover, process and measurement noise are uncorrelated. The full state vector is reconstructed through an observer as in Eq. (10.111). It can be proved that the following observer gain matrix:

$$\mathbf{L} = \mathbf{P}_{\delta\mathbf{x}} \mathbf{C}^T \mathbf{P}_w^{-1}, \quad (10.118)$$

with $\mathbf{P}_{\delta\mathbf{x}} = E\{\delta\mathbf{x}\delta\mathbf{x}^T\}$ satisfying the Riccati equation:

$$\mathbf{AP}_{\delta\mathbf{x}} + \mathbf{P}_{\delta\mathbf{x}} \mathbf{A}^T - \mathbf{P}_{\delta\mathbf{x}} \mathbf{C}^T \mathbf{P}_w^{-1} \mathbf{CP}_{\delta\mathbf{x}} + \mathbf{P}_v = 0, \quad (10.119)$$

minimizes the observation mean square error. This optimal observer is known as Kalman–Bucy filter. Note that there is an analogy between the optimal observer and the optimal regulator. Eq. (10.119) is analogous to Eq. (10.109), with $\mathbf{A}^T \leftrightarrow \mathbf{A}$, $\mathbf{C}^T \leftrightarrow \mathbf{B}$, $\mathbf{P}_{\delta\mathbf{x}} \leftrightarrow \mathbf{S}$, $\mathbf{P}_w \leftrightarrow \mathbf{Q}$, and $\mathbf{P}_v \leftrightarrow \mathbf{R}$.

The control law resulting from the combination of optimal observer and LQR is called linear quadratic Gaussian control, and it is:

$$\mathbf{u} = -\mathbf{K}\tilde{\mathbf{x}} = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{S}\tilde{\mathbf{x}} \quad (10.120)$$

where $\tilde{\mathbf{x}}$ is obtained from the estimator in Eq. (10.111) with the gain matrix in Eq. (10.118).

Adaptive control

Among several modern control strategies that can be applied to satellites control, an interesting one is adaptive control. With adaptive control, it is intended a control paradigm where the controller is modified in real time to fulfill some tasks and respond to the real system behavior.

The most common application of adaptive control sees a fixed controller structure with weights that are changed as the system responds to its inputs. In general, such controllers are nonlinear and may prove difficult to analyze.

There are many categories of adaptive controllers, and a complete overview is out of the scope of the book. However, a brief introduction to the

first two basic forms alongside with the limitations and challenges related to these controllers is presented. The key concept in adaptive controllers is the ability to control a system and trying to make the closed-loop system respond like a reference model. In this way, it is possible to select the response time, steady-state error, and other parameters and design the controller in such a way that the closed-loop real system behaves in the desired way.

The first two basic forms are the Model Reference Adaptive Control (MRAC) and the Adaptive Dynamical Inversion (ADI), which can be seen, depending on the classification, as an indirect MRAC. The first approach is meant to adapt the control gains to follow the reference model while the second is meant to estimate the parameters of the system and follow the reference model.

First, let's define a linear scalar reference model that the adaptive controllers are going to follow:

$$\dot{x}_m = a_m x_m + b_m r, \quad (10.121)$$

where the state is x_m , the reference r , and the system parameters a_m and b_m . We will use a scalar system for clarity; however, this can be generalized for larger scale systems. If we want the system state to converge to the reference, the parameters can be set as $a_m = -b_m = -\lambda$ and the first order linear system is obtained:

$$\dot{x}_m = \lambda(r - x_m), \quad (10.122)$$

which is a simple lowpass filter of the first order with cutoff equal to λ if $\lambda > 0$. Now let the real system be:

$$\dot{x} = a x + b u, \quad (10.123)$$

where u is the control action. The error that the controllers will try to minimize is defined as:

$$e = x - x_m. \quad (10.124)$$

Model reference adaptive control

In the MRAC paradigm, the goal is to estimate the control weights that allow the real system to track the desired system. The controller form is as follows:

$$u = \hat{k}_x x + \hat{k}_r r. \quad (10.125)$$

Substituting in the real system, we get:

$$\dot{x} = (a + b\hat{k}_x)x + (b\hat{k}_r)r. \quad (10.126)$$

Hence, the error would vanish when $a + b\hat{k}_x = a_m = -\lambda$ and $b\hat{k}_r = b_m = \lambda$. If a and b are perfectly known, there is the need to use an adaptive controller; however, the real system properties might be difficult to estimate, prone to error, or changing in time. The goal of adaptive controllers is to face such problematics. In MRAC terms, the estimated control gains should evolve in time to match the theoretical best gains, defined as follows:

$$\begin{cases} k_x = -\frac{\lambda + a}{b} \\ k_r = \frac{\lambda}{b} \end{cases}, \quad (10.127)$$

that are function of the real system variables a and b . If we substitute this into the error dynamics, the following equation is obtained:

$$\dot{e} = -\lambda e + b\Delta k_r r + b\Delta k_x, \quad (10.128)$$

where $\Delta k_r = \hat{k}_r - k_r$ and $\Delta k_x = \hat{k}_x - k_x$. Indeed, if these are null, the error falls to zero according to λ . Another important aspect to consider is the time evolution of the adaptive gains. A Lyapunov function can be constructed, derived with respect to time, and then set to be negative. This procedure is often used in adaptive controllers to guarantee stability and find the parameters update laws. Let's define the function as follows:

$$L = \frac{1}{2}e^2 + \frac{1}{2}|b|\left(\frac{\Delta k_r^2}{\gamma_r} + \frac{\Delta k_x^2}{\gamma_x}\right), \quad (10.129)$$

where the constant positive weights γ_r and γ_x set the rate of adaptation of the controller gains. From Eq. (10.129), the function is always positive and increasing with its variables (i.e., the three errors). Deriving the function and assuming that the ideal gains are static $(\Delta \dot{k}_x \approx \dot{\hat{k}}_x)$, we get that to use Barbalat's lemma and prove stability, the control gain update must be:

$$\begin{cases} \dot{\hat{k}}_x = -\gamma_x e x \operatorname{sign}(b) \\ \dot{\hat{k}}_r = -\gamma_r e r \operatorname{sign}(b) \end{cases}, \quad (10.130)$$

which would give $\dot{L} = -\lambda e^2 < 0$. It should be noted that the sign of the mapping of the control action to the real state must be known beforehand, otherwise the MRAC does not converge. This is not surprising since most of the classical controllers require this minimum knowledge.

Adaptive dynamical inversion

The ADI for of an adaptive controller is meant to directly estimate the system parameters a and b instead of the control gains. The control law then becomes:

$$u = \frac{1}{\hat{b}} ((a_m - \hat{a})x + b_{mr}) = \frac{1}{\hat{b}} (-(\lambda + \hat{a})x + \lambda r). \quad (10.131)$$

The reader should see the analogies with the MRAC control law and its parameters. Following the same procedure discussed in the previous section, the derivative of the error is obtained as:

$$\dot{e} = -\lambda e - \Delta ax - \Delta bu, \quad (10.132)$$

where $\Delta a = \hat{a} - a$ and $\Delta b = \hat{b} - b$. The parameters of the real system, like for the MRAC case, are fixed in time. Again, a Lyapunov function is selected as:

$$L = \frac{1}{2}e^2 + \frac{1}{2} \left(\frac{\Delta a^2}{\gamma_a} + \frac{\Delta b^2}{\gamma_b} \right), \quad (10.133)$$

with adaptive rate gains γ_a and γ_b . Letting the derivative to be negative, as before, the parameter estimation is:

$$\begin{cases} \dot{\hat{a}} = \gamma_a e \\ \dot{\hat{b}} = \gamma_b e u. \end{cases} \quad (10.134)$$

The main difference with the MRAC is that in ADI attention must be paid to sign crossing of \hat{b} , since the control action tends to infinity as $\hat{b} \rightarrow 0$.

Both controllers depend on the state, error, and reference or control action. The error is computed comparing the estimated state with the reference state; hence, it is always needed to propagate the reference model based on the current input reference. This process is computationally more expensive than fixing the controller in the first place, but it can increase the applicability range and even the system robustness.

Additional parameters estimation

Adaptive controllers can also be used to estimate system parameters. Defining the new system as:

$$\dot{x} = ax + bu + \vartheta^T \Phi(x), \quad (10.135)$$

with ϑ and $\Phi(x)$ being the parameters and their basis functions, respectively. The control laws and the updates can be derived as follows.

For the MRAC:

$$\begin{cases} \dot{u} = \hat{k}_x x + \hat{k}_r r + \hat{\vartheta}^T \Phi(x); \\ \dot{\vartheta} = -\Gamma_\vartheta e \Phi(x) \operatorname{sign}(b) \end{cases} \quad (10.136)$$

while for the ADI:

$$\begin{cases} u = \frac{1}{b} \left(-(\lambda + \hat{a})x + \lambda r - \hat{\vartheta}^T \Phi(x) \right) \dot{\vartheta} = \Gamma_\vartheta e \Phi(x). \end{cases} \quad (10.137)$$

The nominal gains/parameters updates are kept equal.

Convergence of parameters

Both ADI and MRAC solutions, as other adaptive paradigms, do not guarantee convergence of parameters unless a persistent excitation is applied on the system. Even in cases where perfect measurements are available, the estimated parameters converge to their true values only with specific classes of reference signals.

Stability does not imply convergence to the real values, but a bounded convergence is expected. The result is expected as estimating parameters of a dynamical system requires a reference signal capable of exciting a response dependent on the said parameters. This applies also to adaptive controls where a state observer is used to estimate the uncertain parameters.

In general, it is not wise to use an adaptive controller to estimate the system parameters, but this information is not always needed in order to have a stable system with prescribed performance.

Adaptive control issues

Both ADI and MRAC parameter update equations rely on the state determination. Since measurements are always affected by noise, it is straightforward to see that noise can negatively affect the variation of the parameters;

hence, a common solution is to apply a dead-zone to the error. In this case, the tracking error e will never reach zero but will converge to a bounded zone. The limit of the dead-zone should be determined taking into account the noise content of the measured state. In general, the source of instability at higher frequencies is a static nonmodeled bias.

It is important to underline that, like classical techniques, also adaptive control can suffer from windup effect. In fact, the parameters are estimated through an integration process. Classical antiwindup techniques should be adopted while applying adaptive control.

Robust control

When the system is too complex to analyze (and simplification with uncertainties are necessary) or has uncertainties that are not easily solved by gain scheduling or other classical approaches, a robust control technique should be adopted to achieve satisfactory performance.

Contrary to adaptive controllers, in robust control, the gains are evaluated a priori and usually do not change adapting to the measures and response of the system. Hence, the problematics of adaptive controllers are avoided. The first introduced methodology is the H_∞ (H infinity) control that can be implemented also in a structured approach. In this case, H_∞ tuning techniques, like loop shaping, are applied to determine the weights of a PID controller. The second is the μ synthesis that uses H_∞ paradigm specializing it for uncertainties in MIMO systems.

H-infinity

The H_∞ (H -infinity) methods can determine a controller with prescribed performances, also in the frequency domain. The name comes from the norm of a transfer function that belongs to the Hardy space (H_∞) and is defined as follows:

$$\|G(j\omega)\|_\infty = \sup_{\omega \in \mathbb{R}} \left(\max_i \lambda_i(G(j\omega)G(j\omega)^T) \right). \quad (10.138)$$

In simpler terms, it is defined as the least upper bound over all real frequencies ω of the Euclidean norm of $G(j\omega)$, here expressed as the maximum singular value of a transfer function matrix $G(j\omega)$. Let us stress the fact that this norm must be computed, at least theoretically, over the whole frequency spectrum.

First, let's consider a linear system plant of state \mathbf{x} , measures \mathbf{y} , control action \mathbf{u} , and performance \mathbf{z} . Performances must be written in a way that

the control goal is to minimize them and is up to the designer to shape them according to requirements.

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{A}(j\omega)\mathbf{x} + \mathbf{B}(j\omega)\mathbf{u} \\ \mathbf{y} = \mathbf{C}_y(j\omega)\mathbf{x} + \mathbf{D}_y(j\omega)\mathbf{u}, \\ \mathbf{z} = \mathbf{C}_z(j\omega)\mathbf{x} + \mathbf{D}_z(j\omega)\mathbf{u} \end{cases} \quad (10.139)$$

where the frequency dependency has been explicitly considered. We can then enforce a control law in the form:

$$\mathbf{u} = \mathbf{K}(j\omega)\mathbf{y}. \quad (10.140)$$

Then, we would have the performance expressed as follows:

$$\mathbf{z} = (\mathbf{C}_z + \mathbf{D}_z\mathbf{K}(\mathbf{I}_n - \mathbf{D}_y\mathbf{K})^{-1}\mathbf{C}_y)\mathbf{x} = \mathbf{F}_t(j\omega)\mathbf{x}, \quad (10.141)$$

where \mathbf{z} is function of a frequency-dependent matrix often called Linear Fractional Transformation (LFT) and the state. For this example, we have assumed to have null reference and the principal task is to bring to zero the state, but, of course, the same approach can be generalized without any issue.

Then the goal is to find $\mathbf{K}(j\omega)$ such that $\|\mathbf{F}_t(j\omega)\|_\infty$ is minimized. This transforms the control problem into an optimization problem. Unconstrained optimization could potentially lead to complex forms of $\mathbf{K}(j\omega)$, while structured version would possibly reach poorer minimization of $\|\mathbf{F}_t(j\omega)\|_\infty$ in absolute value. On the other hand, nonstructured H_∞ may exhibit a complex structure not well suited for online applications; hence, order reduction is often used to derive suboptimal solutions.

The keen reader should have already seen that minimizing \mathbf{F}_t de facto minimizes the influence of the state over the performance; hence, if the system is stable, so will be the performances. There is an underlying assumption that $\|\mathbf{F}_t(j\omega)\|_\infty < 1$ in order to have stability as well.

Up to now, the discussion was not related nor referenced to robustness, but only to performance and state. The extension is quite trivial and shows the real application of H_∞ , with some of its inherent limitations. Let us modify the system dynamics to account for (unmodeled) disturbances, \mathbf{w} :

$$\dot{\mathbf{x}} = \mathbf{A}(j\omega)\mathbf{x} + \mathbf{B}(j\omega)\mathbf{u} + \mathbf{B}_w(j\omega)\mathbf{w}. \quad (10.142)$$

The performance would be now expressed as a function of the disturbances:

$$\begin{aligned} \mathbf{z} &= \left(\mathbf{C}_z + \mathbf{D}_z \mathbf{K} (\mathbf{I}_n - \mathbf{D}_y \mathbf{K})^{-1} \mathbf{C}_y \right) \\ &\quad \left(j\omega \mathbf{I}_n - \mathbf{A} - \mathbf{B} \mathbf{K} (\mathbf{I} - \mathbf{D}_y \mathbf{K})^{-1} \mathbf{C}_y \right)^{-1} \mathbf{B}_w \mathbf{w} = \mathbf{F}_t(j\omega) \mathbf{w}. \end{aligned} \quad (10.143)$$

Minimizing $\|\mathbf{F}_t(j\omega)\|_\infty$ guarantees that, over all possible frequencies, the performance \mathbf{z} is not (less) influenced by the disturbances \mathbf{w} . If \mathbf{F}_t is an appropriate LFT, it is sufficient to have $\|\mathbf{F}_t(j\omega)\|_\infty < 1$ to achieve stability. Note that, uncontrollable or unobservable system would not easily pass this test.

The computation of the norm or the values of $\mathbf{K}(j\omega)$ is out of the scope of the book, but a detailed derivation can be found in. On the other hand, it is important to focus on $\mathbf{B}_w(j\omega)$ that maps the disturbances to the state dynamics. The derivation of $\mathbf{B}_w(j\omega)$ requires a good knowledge of the system and greatly affects the determination of the control law. It should be stressed that a poor modeling would lead to undesirable results. Moreover, measurement noises or actuators errors can be easily included in the disturbances term.

The structured version of the H_∞ has been already used for space applications in the notable example of the Rosetta mission where it has been applied for thruster maneuvers after a failure in the propulsive subsystem had been found.

Mu-control

Mu-synthesis can be seen as an extension of the H-infinity synthesis for MIMO plants with statical or dynamical uncertainties. The name refers to the letter often used to represent the H-infinity robust performance and is a key figure in understanding the effects of the uncertainties in the closed-loop system.

The μ term gives an indication of the effects of the uncertainties on the system output. for a given μ , the peak gain of the system transfer function stays below μ (in normalized units) for uncertainties up to $1/\mu$ of the uncertainties introduced in the transfer function. In general, a value below 1 is preferred. If the analysis shows μ extremely high or even infinity, it means that the system, with the current gains, is not able to guarantee stability in the full range of the considered uncertainties.

Like the H-infinity case, even μ is a function of frequency; hence, the peak value in the frequency domain shall be considered. Even with modern algorithms, it is not easy to compute the exact value in the whole frequency domain, and, often, minimum and maximum boundaries are the only

quantities that can be obtained. Still, important information can be extracted through this method.

One of the most popular ways to use mu-synthesis is the D-K iteration algorithm consisting in iterations of h-infinity and μ -evaluation.

First, the control gain \mathbf{K} ($\mathbf{u} = \mathbf{Ky}$) over the nominal system with no uncertainties is estimated using h-infinity technique, making sure it minimizes the closed-loop gain.

Then, the robust performance μ , according to the uncertainties estimated in the plant, is estimated. The value of μ is then scaled (D-scaling) as the scaled h-infinity performance. The obtained quantity is a function of the uncertainties and of the control gain \mathbf{K} .

Finally, the K-step, as the name suggest, aims at finding a new \mathbf{K} that minimizes the scaled h-infinity performance computed in the previous step.

The process is repeated numerically until convergence to a stable condition.

Since the algorithm is a continuous iteration of h-infinity plus evaluation, if not constrained, it can generate complex gain structures. For this reason, also for the mu-control, it is possible to adopt a structured approach, thus simplifying the controller structure. This is an advantageous solution for real-time controller used in context where computational power is limited, like on-board of a spacecraft.

Robust adaptive controllers

Robust controllers permit to generate control laws that are robust to disturbances and uncertainties in the system, often resulting in complex and computationally expensive algorithms. Sometimes, gains are computed beforehand, and, in general, they are not modified during operations, resulting in cases where the consumption is higher than needed. On the other hand, adaptive controllers can react in real time to changes in the model but are not specifically targeted to disturbance or uncertainties rejection as they usually force the system to follow a predetermined behavior.

A new class of adaptive controllers can be defined as those controllers that modifies the gains (or the structure) of the control law to guarantee robustness in cases where the system is uncertain or complex (bounded) disturbances are present.

Typically, this can be achieved by a Lyapunov function, where the process must take into account disturbances and uncertainties and, in the end, will be able to relate the gain update in time with the state of the system. The process is quite similar to the one used for adaptive controllers, but the aim is the

one of the robust controllers. There are several examples in literature of this kind of controllers, each with a different approach or way to execute the task or to counteract specific uncertainties. See for a broader overview.

Model predictive control

The review of the most common control methods has covered important topics such as model-free feedback control, unconstrained optimal regulator, and adaptive control. The benefits and drawbacks of the mentioned approaches have been thoroughly discussed in the previous sections. MPC represents a synthesis of the above strategies: it is an optimization-based control scheme that merges the advantage of constrained optimization and feedback control. Due to its optimization-based foundations, MPC is often used to synthesize guidance, coupled with a low-level controller to track the desired control profile.

The rationale behind this type of control lies on the inherent limitations of offline optimal control. By definition, the classical optimal control problem is based on an analytical cost function and dynamical model, which are used to determine and plan the control sequence to be executed. Obviously, the more the dynamical model is adherent to reality, the more we can be confident that the offline control sequence, if executed in real operations, will drive the spacecraft to the desired target state, optimizing the cost function. Sticking to the familiar terms introduced in the chapter, the optimal control approach does not present any feedback, i.e., it can be seen as an open-loop controller. Nevertheless, we understood how feedback is a critical feature for the effective control of a generic plant. MPC is essentially a constrained receding horizon optimization approach to plan control actions following optimal control problems solved sequentially and iteratively at each timestep Δt_{MPC} .

The optimization covers N_s time steps, which results in N_s control inputs, or N_m if the control horizon is shorter. In rigorous terms, N_s denotes the length of the prediction horizon or output horizon, and N_m denotes the length of the control horizon or input horizon ($N_m \leq N_s$), as shown in Fig. 10.16. When $N_s = \infty$, we refer to this as the infinite horizon problem, and similarly, when N_s is finite, as a finite horizon problem. Only the first computed control is executed, and, at the next time step, the planner solves the optimization problem again based on the actual current state. In its simplest form, the objective function used in MPC seeks the minimization of the quadratic difference between the target state, $\tilde{\mathbf{x}}$, and the plant state, \mathbf{x}_k

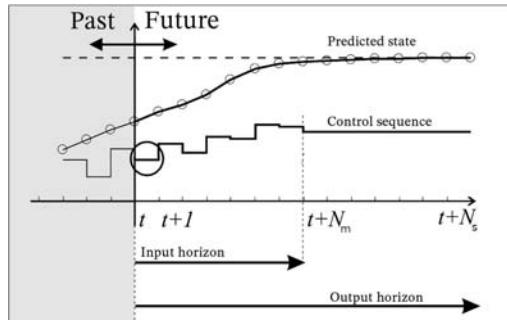


Figure 10.16 Receding horizon scheme: only the first control action is executed.

at each time step, and a quadratic term representing the control effort, similarly to Eq. (10.105) here reported for a generic control objective not limited to regulation:

$$\mathcal{J} = \int_0^{t_f} \left(\mathbf{x} - \tilde{\mathbf{x}} \right)^T \mathbf{Q} \left(\mathbf{x} - \tilde{\mathbf{x}} \right) + \mathbf{u}^T \mathbf{R} \mathbf{u} dt + \left(\mathbf{x}_f - \tilde{\mathbf{x}}_f \right)^T \mathbf{Z} \left(\mathbf{x}_f - \tilde{\mathbf{x}}_f \right)$$

where \mathbf{Q} , \mathbf{R} , and \mathbf{Z} are weighting symmetric matrices, with \mathbf{Q} and \mathbf{Z} positive semidefinite and \mathbf{R} positive definite, and $\mathbf{x}_f = \mathbf{x}(t_f) = \mathbf{x}_{k+N}$, with $t_f = N_s \cdot \Delta t_{MPC}$ being the final time of the interval of interest, also known as the prediction horizon (Fig. 10.17). For practical application and

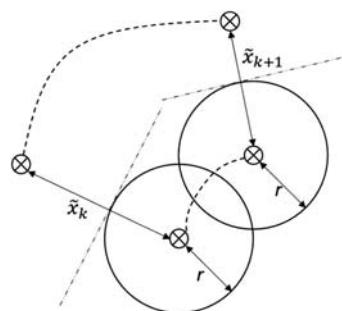


Figure 10.17 Example of collision avoidance constraint.

solution, optimal control is casted into linear or nonlinear programming. For this reason, the discrete form is much more useful and reads:

$$\begin{aligned} \mathcal{J}(\mathbf{x}_k, \mathbf{u}_k) = & \left(\mathbf{x}_{k+N} - \tilde{\mathbf{x}}_k \right)^T \mathbf{Z} \left(\mathbf{x}_{k+N} - \tilde{\mathbf{x}}_k \right) \\ & + \sum_{i=1}^{N-1} \left(\mathbf{x}_{k+i} - \tilde{\mathbf{x}}_k \right)^T \mathbf{S} \left(\mathbf{x}_{k+i} - \tilde{\mathbf{x}}_k \right) \\ & + \sum_{i=0}^{N-1} \mathbf{u}_{k+i}^T \mathbf{R} \mathbf{u}_{k+i} \end{aligned} \quad (10.146)$$

where \mathbf{Z} is the weighting (or penalty) on the final time step. The weighting on the final state variation \mathbf{Z} is formed from the discrete-time algebraic Riccati equation, as in the unconstrained LQR, and it is typically enforced to guarantee the stability of the closed-loop system.

The overall performance of an MPC is strongly linked with the selection of its three fundamental parameters. The specific numerical values are strongly dependent on the mission requirements and scenario. Nevertheless, we can draw insights to help in the MPC tuning process.

- Prediction horizon. The prediction (or receding) horizon $t_f = N_s \cdot \Delta t_{MPC}$ defines how far ahead in the future the controller seeks to optimize the evolution of the controlled dynamics. In general, a low value yields a controller unable to take advantage of the long-term dynamics evolution to find the optimal control. On the contrary, a large prediction horizon leads to an excessive computational time that may be unnecessary in terms of optimality, jeopardizing the actual feasibility of the on-board implementation.
- Sampling time. The sampling time Δt_{MPC} defines the time span between two successive optimizations with feedback signals or states. A small value is always desirable in terms of accuracy, as a faster update can better represent the dynamics evolution and to modulate the control action more precisely if needed. However, keeping constant the prediction horizon, a smaller sampling time implies a larger dimension of the optimization vector. Since the optimization computational time depends on the problem size, a small sampling time may be prohibitive for on-board implementation.
- Control horizon. The control horizon indicates for how many sampling steps the control resulting from the solved optimal problem is applied, before restarting the optimization to update the control profile. The

most robust option is to recompute the control every sampling time, to deal with unpredicted changes in the external conditions as soon as possible. However, longer control horizons relax the frequency at which the on-board computer needs to recompute the control profile, relieving computational effort that could be allocated to other tasks.

The dynamical model functions as a constraint between successive plan states. Hence the optimal control problem reads:

$$\min_{\mathbf{u}} \mathcal{J}(\mathbf{x}_k, \mathbf{u}_k)$$

$$\text{subject to } \mathbf{x}_{k+i+1} = f(\mathbf{x}_{k+i}, \mathbf{u}_{k+i}), \quad i = 1, \dots, N_s \quad (10.147)$$

where the dynamics is encoded into the constraint, in which each state of the receding horizon is linked to the previous one in the sequence, i.e., the pedix. The cost function minimization may be subject to different constraints. The most common in space applications are hereby listed:

- *Actuator control action constraint.* Typically, maximum or minimum admissible control is enforced by constraining the vector value within a convex domain, defined as:

$$\mathbf{u}_{\min} < \mathbf{u}_i < \mathbf{u}_{\max}$$

It is important to remark that this is a convex constraint due to its vectorial nature. The domain is convex entailing a hypercube with no holes. This constraint is conceptually different from limiting the minimum and maximum absolute value of the actuation [14].

To understand this, let us go through an example: imagine we have a robot with 1 degree of freedom to be controlled with a maximum acceleration $1 \frac{\text{m}}{\text{s}^2}$ and a maximum deceleration $-0.5 \frac{\text{m}}{\text{s}^2}$. Then we would enforce the convex control action constraint as $-0.5 \frac{\text{m}}{\text{s}^2} < u_i < 1 \frac{\text{m}}{\text{s}^2}$. The lower bound is used to define the negative value. There is only no limit on the minimum impulse the throttle can deliver. Let us now imagine the situation in which the maximum acceleration and deceleration absolute value limit is $1 \frac{\text{m}}{\text{s}^2}$, but the minimum acceleration and deceleration we can get is constrained to $10^{-4} \frac{\text{m}}{\text{s}^2}$, absolute value. This means that the acceptable control would be: $-1 \frac{\text{m}}{\text{s}^2} < u_i < -10^{-4} \frac{\text{m}}{\text{s}^2}$ or $1 \frac{\text{m}}{\text{s}^2} < u_i < 10^{-4} \frac{\text{m}}{\text{s}^2}$, which is not a convex constraint.

- *Collision avoidance constraint.* The collision avoidance constraint refers to the imposition of given distance from a certain static or moving entity throughout the optimization path. The easiest way to picture this constraint is to think of a trajectory that needs to keep a certain minimum

distance from an obstacle, such as another spacecraft. Unfortunately, this constraint is concave; hence, it must be treated carefully (see use case scenario in [Chapter 14—Applicative GNC cases and examples]).

$$1 - (\mathbf{x}_{k+i} - \mathbf{x}_{k+i}^o)^T \mathbf{C}^T \mathbf{P} \mathbf{C} (\mathbf{x}_{k+i} - \mathbf{x}_{k+i}^o) < 0, i = 1, \dots, N$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_3 \\ \mathbf{0}_3 \end{bmatrix}$$

where the superscript o refers to the *keep-out-zone* is an ellipsoid centered on the obstacle and expressed by its quadratic form with the positive semi-definite matrix \mathcal{P} . If the *keep-out-zone* is a sphere, the quadratic form is simply a symmetric matrix $\mathcal{P} = r^{-2} \mathbf{I}_3$ where r is the spherical safe region radius.

Up to now, it is evident how MPC resembles the LQR scheme. Indeed, even though MPC can optimize a generic cost function, it is true that, in its simplest form, MPC can be thought as a constrained version of the classical LQR with finite time horizon. Indeed, as described for the LQR, the cost function describes mathematically the design trade-off between the control effort and the target reaching performance. If the target state is penalized more than the control, the closed-loop system will move rapidly toward the target state, but it will require a high control effort. On the other hand, if the quadratic term involving the input is larger than the one of the target states, the control energy will be limited, but the plant may take long time to reach the target state.

Again, diagonal matrices are used as weights. The diagonal elements describe how much each state and control input contribute to the overall cost. When a particular distance to target state or control input needs to be penalized, the associated diagonal element is chosen larger than the other ones. For this reason, it is also common to normalize the weights choosing the identity matrix as one of the weights (\mathbf{Q} or \mathbf{R}).

The MPC algorithms mainly differ for the selection of the objective function and the different dynamical models. Since MPC is a control strategy designed to be executed on-board, the applications strive to reduce the computational complexity of the finite-time receding horizon optimization. The most common MPC alternatives, in terms of objectives and constraints, are reported in [Table 10.3](#):

Table 10.3 MPC algorithms alternatives.

Cost function	Dynamical model	Constraints	Solution
Norm-1 linear	Linear	Linear	Linear programming
Quadratic	Linear	None	Explicit (cfr. LQR)
Quadratic	Linear	Linear	Quadratic programming
Nonlinear	Nonlinear	Nonlinear	Nonlinear programming

Let us go through an example on how to design a simple MPC for relative trajectory control or rendezvous. We want to design an MPC scheme that controls the trajectory of a spacecraft with respect to a target in nearly circular orbits. The cost function we want to optimize is quadratic on the control effort and on the distance to the final state. The spacecraft is subject to the presented Clohessy–Wiltshire dynamical model (see [Chapter 4 – Orbital dynamics – Relative dynamics](#)) and has a maximum thrust constraint.

In order to solve efficiently the constrained optimal control problem, we seek to express the whole cost function as a function of the input control, which represents the decision variable of the optimization. If the collision avoidance constraint is not enforced, the MPC reduces to a linearly constrained quadratic optimization problem. Hence, the problem can be recast into Quadratic Programming formulation, using [Eq. \(10.146\)](#) as basis. The quadratic programming formulation for the MPC can be rewritten as:

$$\min_{\mathbf{U}_k} \frac{1}{2} \mathbf{U}_k^T \mathbf{Q} \mathbf{U}_k + \mathbf{H} \mathbf{U}_k$$

$$\text{subject to } \mathbf{V} \mathbf{U}_k < \mathbf{W}$$

where $\mathbf{U}_k = [\mathbf{u}_k, \dots, \mathbf{u}_{k+N_s-1}]$ is a stacked vector containing the decision variables for the optimization problem, namely the control action for each discretization time step. The dynamics is inserted directly into the cost function using the linearized Clohessy–Whiltshire model and its state transition matrix (cfr. [Chapter 4 – Orbital Dynamics – Relative Dynamics](#)). Indeed, one can write the compact form using the state transition matrix and comprising the stacked vectors for all the N_s time instants:

$$\mathbf{X}_k = \boldsymbol{\Psi} \mathbf{x}_k + \boldsymbol{\Omega} \mathbf{U}_k$$

where the relevant matrices can be derived from mathematical manipulations of the stacked form of Eqs. (10.146)–(10.147):

$$\boldsymbol{\Psi} = [\boldsymbol{\Phi}(t_{k+1}, t_k), \boldsymbol{\Phi}(t_{k+2}, t_k), \dots, \boldsymbol{\Phi}(t_{k+N_s}, t_k)]^T$$

$$\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\Phi}(t_{k+1}, t_k)\mathbf{B} & 0 & \cdots & 0 \\ \boldsymbol{\Phi}(t_{k+2}, t_k)\mathbf{B} & \boldsymbol{\Phi}(t_{k+2}, t_{k+1})\mathbf{B} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\Phi}(t_{k+N_s}, t_k)\mathbf{B} & \boldsymbol{\Phi}(t_{k+N_s}, t_{k+1})\mathbf{B} & \cdots & \boldsymbol{\Phi}(t_{k+N_s}, t_{k+N_s-1})\mathbf{B} \end{bmatrix}$$

and

$$\mathbf{Q} = 2\mathcal{L}_1 + 2\boldsymbol{\Omega}^T \mathcal{L}_2 \boldsymbol{\Omega}$$

$$\mathcal{H} = 2\mathbf{x}_k^T \boldsymbol{\Psi}^T \mathcal{L}_2 \boldsymbol{\Omega} - 2\boldsymbol{\Gamma}_k^T \mathcal{L}_3 \boldsymbol{\Omega} - 2\tilde{\mathbf{x}}_k^T \mathbf{Z} \mathcal{L}_4 \boldsymbol{\Omega}$$

$$\mathcal{L}_1 = \begin{bmatrix} \mathbf{R} & 0 & \cdots & 0 \\ 0 & \mathbf{R} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{R} \end{bmatrix} \in \mathcal{R}^{3N_s \times 3N_s},$$

$$\mathcal{L}_2 = \begin{bmatrix} \mathbf{S} & 0 & \cdots & 0 \\ 0 & \mathbf{S} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{Z} \end{bmatrix} \in \mathcal{R}^{6N_s \times 6N_s}$$

$$\mathcal{L}_3 = \begin{bmatrix} \mathbf{S} & 0 & \cdots & 0 \\ 0 & \mathbf{S} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathcal{R}^{6N_s \times 6N_s},$$

$$\mathcal{L}_4 = [0 \quad 0 \quad \cdots \quad \mathbf{I}_6] \in \mathcal{R}^{6 \times 6N_s}$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{U}_{\max} \\ \mathbf{U}_{\min} \end{bmatrix} \in \mathcal{R}^{6N_s \times 1},$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{I}_{3N_s} \\ -\mathbf{I}_{3N_s} \end{bmatrix} \in \mathcal{R}^{6N_s, 3N_s}$$

Finally, the vector $\mathbf{\Gamma}_k$ is a stacked vector containing the desired state for each time step of the receding horizon (it can be fixed to the final position or moving along a trajectory). The problem in this form can be solved at each time step with different algorithms falling within the large domain of function minimization, e.g., interior point, gradient descent, etc.

Robust model predictive control

Although MPC optimizes the control sequence repeatedly, its efficacy is largely influenced by the accuracy of the dynamical model representation. Robust variants of MPC try to include bounded disturbance while minimizing the cost function and reaching the objectives. A nonexhaustive list of robust approaches to MPC is given below. Essentially, they differ in the representation of the uncertainties affecting the system [15].

Different uncertainty sets have been proposed in the literature in the context of MPC and are mostly based on time-domain representations. Frequency-domain descriptions of uncertainty are not suitable for the formulation of robust MPC because MPC is primarily a time-domain technique. A nonexhaustive list of techniques for robust MPC is reported here [15]:

- Impulse/step response.
- Structured feedback uncertainty.
- Multiplant description.
- Bounded input disturbances.

The exhaustive discussion on these advanced methods is left to the reader [15], since it is beyond the scope of this book.

Sliding mode control

SMC is a nonlinear control method that computes the control action from a set of admissible functions. The objective of the control is to drive the system to slide on a surface, defined in the phase space, which is representing the system's target behavior. The state feedback control law is not a continuous function of time, but it switches from a continuous control structure to another one according to the system state with respect to the sliding surface (i.e., if the states are above or below the surface). Thus, SMC can be categorized in the class of the variable structure control methods.

To visualize the concept of SMC, we can imagine a relay-based control system, where the relays are electrically operated switches that are used to implement logical on-off control actions. The on-off controls can be regarded as primordial variable structure control strategies, which established the theory behind SMC.

A generic SMC operation can be divided in two distinct phases: reaching phase and sliding phase. The former sees the system approaching the sliding surface by exploiting one continuous control structure (e.g., a switch that is always on, in the relay-based simplified analogy). The latter takes place when the sliding surface is reached. Then, the discontinuous controller forces the states to slide toward the set point (i.e., the origin in the phase space). This behavior is shown in Fig. 10.18.

Let's consider a second-order nonlinear system:

$$\ddot{x} = f(x, \dot{x}) + u,$$

where u is the control input, which is designed to make the system reaching a desired state, x_{ref} . We can define the sliding surface, s , as:

$$s = \Delta \dot{x} + k \Delta x,$$

where k is a scalar and $\Delta x = x - x_{\text{ref}}$ is the control error. If we force the system to stay on the sliding surface, the SMC can be derived by setting $\dot{s} = 0$.

For example, in the problem of spacecraft attitude control, we may define the sliding surface as:

$$s = \delta \omega + k \delta q_{1:3},$$

where $\delta \omega$ is the angular rate error, and $\delta q_{1:3}$ is the vectorial part of the error quaternion.

Imposing $\dot{s} = 0$, we obtain:

$$\begin{aligned} \dot{s} &= \dot{\omega} - \dot{\omega}_{\text{ref}} + \frac{1}{2} k H(\delta q) \delta \omega \\ &= \dot{\omega} - \dot{\omega}_{\text{ref}} + \frac{1}{2} k (\delta q_4 \delta \omega + \delta q_{1:3} \times (\omega + \omega_{\text{ref}})), \end{aligned}$$

where we used the quaternion attitude kinematics rules.

Thus, to control the spacecraft dynamics:

$$\dot{\omega} = -I^{-1}[(\omega \times I\omega) - t],$$

The control torque can be derived as:

$$t = I \left(\dot{\omega}_{\text{ref}} - \frac{1}{2} k (\delta q_4 \delta \omega + \delta q_{1:3} \times (\omega + \omega_{\text{ref}})) \right) + (\omega \times I\omega).$$

The SMC has the advantage of not being very sensible to the uncertainties on system parameters, and it provides stability with respect to modeling unknowns. Moreover, it allows to reduce the modeling order of the dynamics, it guarantees finite convergence time, and it provides a convenient control profile for systems with on-off actuators (e.g., thrusters). In general, SMC offers a simple and robust control action, which is suitable for online real-time applications with limited computing capabilities.

However, its main weakness is related to the chattering, which is due to implementation imperfections. Indeed, the switching control structure makes the system to chatter in a close neighborhood of the sliding surface. This is particularly problematic for highly accurate systems and for spacecraft with flexible appendages and internal sloshing. Chattering can be reduced through the use of dead bands or boundary layers around the sliding surface. In alternative, filtered sliding mode or higher order SMCs may be used to alleviate the chattering effect. Chattering can be visualized along the sliding surface in Fig. 10.18.

Control budgets

Once a spacecraft control design is finalized by using one of the techniques recalled in this chapter, a control budget is needed in order to predict the actual behavior of the spacecraft in closed loop. In particular, there are three ways to characterize a control design:

- Stability of the system in nominal conditions.
- Performance of the system in nominal conditions.

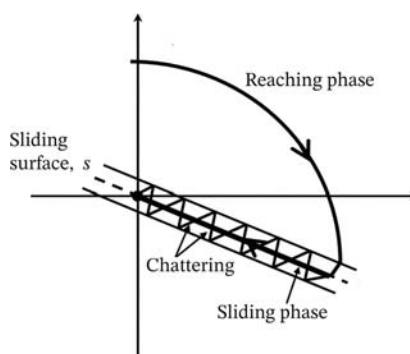


Figure 10.18 Sliding mode control operation. Reaching phase and sliding phase are visible.

- Stability and performance robustness in face of modeled parametric uncertainties and neglected dynamics.

According to stability, common industrial practice is to use the traditional stability margins (gain, phase, delay) to verify the robustness of the closed-loop system. However, the use of these classical tools is justified when a hypothesis of decoupling (or small coupling) among control axis stays and allows the control engineer to deal with axis-by-axis SISO synthesis. This condition is not always verified, especially when large flexible appendages, like solar arrays, are installed on the main spacecraft hub. In these cases, a complex MIMO system has to be considered, for which classical stability margins cannot provide a complete understanding of the system robustness, while the modulus margin can result more appropriate.

When dealing with spacecraft control performance, we generally refer to a set of requirements imposed by the nature of the mission. A LEO observation mission generally demands a high agility (cfr. [Chapter 1](#) – Introduction) in order to guarantee a high ground coverage rate for instance. This means that the satellite has to be able to make fast slew maneuvers that impose to reduce the controller settling time and maximize the available imaging window. On the other hand, when very tight pointing performances are demanded for long observations of outer space objects, like exoplanet, the level of pointing accuracy makes the set of performance even more challenging as for the case of the new generation of space telescope as the James Webb platform, launched in December 2021. For these missions, the same attitude control system can be a source of image perturbation. The reaction wheels used for coarse attitude control can in fact suffer from imbalances, ball bearing, and motor imperfections, as seen in [Chapter 3](#) – The Space Environment – Internal Perturbations, that propagate through the spacecraft structure all the way to the scientific payload, causing an irremediable loss of information in the high frequency bandwidth.

For these reasons, the European Space Agency (ESA) made an effort to overcome the individual industrial inhouse methods and propose common guidelines and practice to evaluate the control pointing budget.

The ECSS Control Performance Standard [16] and the ESA Pointing Error Engineering Handbook [17] define a framework for control performance for any space system, including the space segment, the ground segment, and the launch service segment. These documents can be applied to evaluate the stability and performance of closed-loop algorithms with a set of prescribed indexes. Due to the already evocated importance of pointing performance, most of these documents are based on the definition of the

pointing error. Common practice is to analyze the error at each instant t of a time domain observation, while performance can be defined on a *window time* Δt as well. This is important, for instance, when performance has to be bounded to guarantee a correspondent level of image or communication quality by relying on the performance indexes to the payload integration time.

In the same way, a *stability time* Δt_s can be defined to evaluate the stability of the control algorithms by comparing the relative error on different observations of time window Δt .

Fig. 10.19 shows an example of windowed time and stability time for an observation time span.

Notice that pointing performance and stability can be defined not only for the attitude control system generally related to the spacecraft main body but also for payload elements, like antennas for high-accuracy pointing. In case of large and flexible spacecraft, in fact, flexible appendages connected to the rigid main body could experience a degradation of the pointing requirement due to the excitation of their natural modes.

Let consider now a generic spacecraft for which pointing error has to be determined. Following ESA standards, we can distinguish:

- *Absolute knowledge error* (AKE). Difference between the actual parameter (main body attitude, payload attitude, etc.) and the known (measured or estimated) parameter in a specified reference frame.
- *Absolute performance error* (APE). Difference between the target (commanded) parameter (main body attitude, payload attitude, etc.) and the actual parameter in a specified reference frame.
- *Mean knowledge error* (MKE). Mean value of the AKE over a specified time interval Δt .

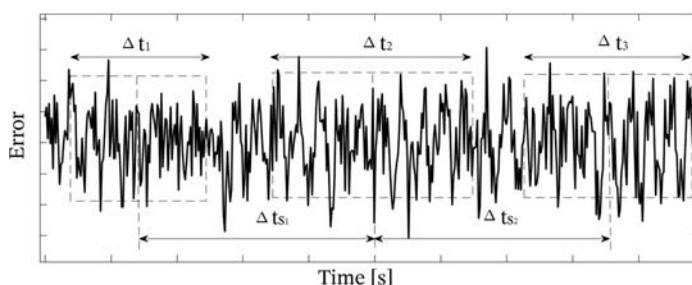


Figure 10.19 Example of window time and stability time for definition of pointing errors [16,17].

- *Mean performance error* (MPE). Mean value of the APE over a specified time interval Δt .
- *Relative knowledge error* (RKE). Difference between the APE at a given time within a time interval Δt and the MKE over the same time interval.
- *Relative performance error* (RPE). Difference between the APE at a given time within a time interval Δt and the MPE over the same time interval.
- *Knowledge drift error* (KDE). Difference between MKEs taken over two time intervals separated by a specified time Δt_s within a single observation period.
- *Performance drift error* (PDE). Difference between MPEs taken over two time intervals separated by a specified time Δt_s within a single observation period.
- *Knowledge reproducibility error* (KRE). Difference between MKEs taken over two time intervals separated by a specified time Δt_s within different observation periods.
- *Performance reproducibility error* (PRE). Difference between MPEs taken over two time intervals separated by a specified time Δt_s within different observation periods.

The definition of these pointing errors allows space control engineers to make accurate prediction of worst-case configurations both in linear and nonlinear domain. In linear domain, frequency analyses are possible by considering the statistical description of all possible error sources (i.e., sensor noises, external orbital perturbations, internal disturbances like solar array drive mechanism driving signal, etc.) and the propagation of the disturbance to the pointing error through the system. If, in fact, an LTI transfer function $H(j\omega)$ of the plant from the error source to the pointing error is available, as for the vast majority of space missions, where the spacecraft rates are generally kept very low, the probability distribution of the pointing error (see [Chapter 12 - GNC Verification and Validation - Statistical methods](#)) is provided by the linear combination of the probability distribution of each error source through the linear plant.

The variance of an error source σ_{es}^2 described as random processes is, in fact, related to its power spectral density (PSD) $P_{es}(\omega)$:

$$\sigma_{es}^2 = \frac{1}{2\pi} \int_0^\infty P_{es}(\omega) d\omega$$

The PSD of the output error $P_{oe}(\omega)$ depends on the transformation of $P_{es}(\omega)$ through the system $H(j\omega)$:

$$P_{oe}(\omega) = |H(j\omega)|P_{es}(\omega)$$

The variance of the output error signal σ_{oe}^2 is thus computed from its PSD $P_{oe}(\omega)$:

$$\sigma_{oe}^2 = \frac{1}{2\pi} \int_0^\infty P_{oe}(\omega) d\omega$$

Moreover, it is possible to translate the pointing indexes defined in time domain also in frequency domain by expressing them as PSD weighting function F_{metric} . Simplified rational approximations \tilde{F}_{metric} , used for control synthesis and analysis, such that $F_{metric}(\omega) \cong |\tilde{F}_{metric}(s)|^2$, where $s = j\omega$, were proposed by Ref. [18] and recalled in Ref. [17]. Table 10.4 resumes the set of rational approximation of pointing weighting functions.

Another performance related to pointing error is the *tranquillization time*. The tranquillization time is generally related to the time duration needed before achieving a specified level of pointing performance or angular rate. An example is provided by the oscillation of a large flexible structure, like a telecommunication antenna, induced by a slew maneuver or a reaction wheel offloading. The attitude control system in this case is forced to reduce this oscillation to an admitted value within the imposed tranquillization time.

Pointing performance is not the unique criterium to take into account for a spacecraft control design. Another important parameter limiting the achievable ideal performances is the *actuation authority*, which means the maximum available command reachable by the set of chosen actuators. For this reason, when designing an attitude control system, for instance, it is important to verify that in the predicted worst-case configuration, the spacecraft will not ask for a control signal bigger than the saturation value of its actuators. If this is the case, a complementary solution has to be found by punctually using an alternative set of actuators. Common ways to desaturate the reaction wheels is to employ magnetometers on CubeSat platforms or chemical thrusters (mainly dedicated to orbital control) on bigger satellites.

Table 10.4 Rational approximations of pointing weighting functions [17,19].

Error index	Metric	Rational weighting function $\tilde{F}_{metric}(s)$
APE	Absolute	$\tilde{F}_A(s, \Delta t) = 1$
RPE	Windowed variance	$\tilde{F}_{WV}(s, \Delta t) = \frac{s\Delta t(s\Delta t + \sqrt{12})}{(s\Delta t)^2 + 6(s\Delta t) + 12}$
MPE	Windowed mean	$\tilde{F}_{WM}(s, \Delta t) = \frac{2(s\Delta t + 6)}{(s\Delta t)^2 + 6(s\Delta t) + 12}$
PDE, PRE	Windowed mean stability	$\tilde{F}_{WMS}(s, \Delta t, \Delta t_s) = \tilde{F}_{WM}(s, \Delta t) \frac{2s\Delta t_s(s\Delta t_s + 6)}{(s\Delta t_s)^2 + 6(s\Delta t_s) + 12}$

Noise coming from sensor measurements or actuator signals should also be attenuated above the control bandwidth in order to not introduce a high frequency undamped error.

All these performances are generally concurrent, and a good compromise among pointing performance, control level, and noise attenuation has to be found. The optimality point is, of course, driven by sensors and actuators quality that bound the achievable ideal performance.

The last point to be checked in a control design is the stability and performance robustness when parametric or dynamical uncertainties affect the system. Parametric uncertainties are generally related to mechanical properties of the plant like a misknowledge on the value of the mass and inertia due to unavoidable mismatch between preliminary and final spacecraft design. Another example is the error made in the computation of the frequencies and damping of the natural modes of the structure based on finite element theory and used for control synthesis. On the other hand, dynamical uncertainties are related to unmodeled dynamics like neglected high frequency dynamics in actuator models.

If the state-space matrices of an LTI can be written as polynomial or rational functions of the uncertainties, an equivalent LFT of the system can be easily derived. Let consider the following partition of the nominal LTI plant $P(s)$:

$$P(s) = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}$$

and Δ being the block of uncertainties. The lower $\mathcal{F}_l(P(s), \Delta)$ and upper LFT $\mathcal{F}_u(P(s), \Delta)$ are defined as:

$$\mathcal{F}_l(P(s), \Delta) = P_{11} + P_{12}\Delta(I - P_{22}\Delta)^{-1}P_{21}$$

$$\mathcal{F}_u(P(s), \Delta) = P_{22} + P_{21}\Delta(I - P_{11}\Delta)^{-1}P_{12}$$

and shown in Fig. 10.20, where \mathbf{u} and \mathbf{y} are the classical system inputs and outputs, while \mathbf{w} and \mathbf{z} are the inputs and outputs to the modeled uncertainties in the Δ block.

The size of the smallest perturbation $\Delta \in \Delta$ which brings a pole of the interconnection on the imaginary axis at frequency ω is:

$$k(\omega) = \min_{\Delta \in \Delta} \{\bar{\sigma}(\Delta) : \det(I - P(j\omega)\Delta) = 0\}$$

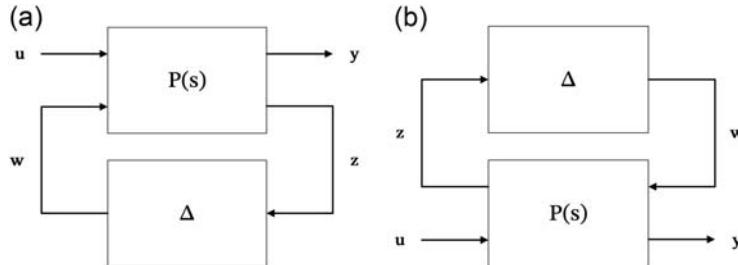


Figure 10.20 Lower (a) and upper (b) LFT standard forms.

with $\bar{\sigma}(\Delta)$ maximum singular value of the Δ block.

The *structured singular value* $\mu_\Delta(P(j\omega))$ is defined as:

$$\mu_\Delta(P(j\omega)) = \frac{1}{k(\omega)}$$

The *robustness margin* is finally defined as the size of the smallest perturbation $\Delta \in \Delta$ which brings a pole of the LFT interconnection on the imaginary axis:

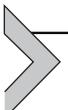
$$M_r = \min_k(\omega) = \min_\omega \frac{1}{\mu_\Delta(P(j\omega))} = \frac{1}{\max_\omega \mu_\Delta(P(j\omega))}$$

The definition of the structured singular value helps in the evaluation of the robustness of both stability and performance of the system by considering in the analysis the right input/output channels of the LFT interconnection.

The biggest advantage of using mu-analysis for performance robustness is to fast detect worst-case scenarios by avoiding time expensive Monte Carlo's simulations that can miss very rare critical configurations.

For an extensive understanding of μ theory, the reader is invited to refer to Ref. [20]. Recent developments of μ theory have enhanced the way of evaluating the stability and system performance of MIMO system with a complex set of parametric and dynamical uncertainties [21].

The final step to evaluate a control budget is to validate the system on a high-fidelity simulator which takes into account all possible nonlinearities not considered in the control synthesis. In this case, Monte Carlo's simulation campaigns are generally the most widespread validation and verification practice across the industry (see Chapter 12 –GNC verification and validation).



Control implementation best practices

The steps to go from algorithms theory to actual software implementation are nontrivial and deserve a special mention. The software implementation is a necessary step for a complete Attitude and Orbit Control System (AOCS) and is meant to connect algorithms to hardware. As we are talking about software, one should always have in mind that there are several standards based on the language the code is written and can be helpful in designing any part of an AOCS software. Here, we will focus on general concepts, while recommendations as language specificity (e.g., C code, ADA code, etc.) or certifications might be taken from general available resources.

First, start from the control functional architecture definition, collecting inputs from requirements and other considerations made at system level. Then, draft the code components and start defining the interfaces with the rest of the AOCS code, as data transfer and availability are paramount.

Always have a full AOCS suite simulation available to test the control scheme with the rest of the system and with the dynamical representation of actuators, sensors, and the satellite as a whole, depending on the accuracy level available at the time of development.

In the development process, add and test all functionalities and build regression tests to be sure that an update is not jeopardizing previously established features.

Especially in the control area, weights management is paramount. Controllers' weights are deeply dependent on the system they are supposed to control; hence, a change in inertia, pointing request, or actuators behavior would likely trigger a reweighting, especially during fast-paced projects. Hence, it is always a good idea to have reduced order models to test the controllers in a fast way before updating weights with a more robust procedure. Another issue that can trigger a reweight is a change in navigation or guidance as this can change the system response in frequency, that is why it is always good to make periodic testing of the whole AOCS suite.

The rule number one for the implementation of an effective control is to reason always in discrete form. There is almost no space application that can run so fast to make the continuous theory work efficiently: always code and reason in discrete form.

Finally, some general remarks to take for code implementation. A useful code is functional, readable, and maintainable, a good code is also optimized, reusable, and configurable/flexible.

The code shall be well documented (comments and algorithm documentation) and organized in a clear way so that more people can contribute to it. Break the code in functional blocks and in smaller functions especially if it is possible to reuse them as in a library of functions. Do not exaggerate in nesting functions, too many levels would make the code unreadable and thus not maintainable.

In this paragraph, some recommendations for the design of a control system are summarized. They have been already introduced along the chapter, but they are schematically reported here to the benefit of the reader.

The very initial step for the control design is the study of the system to be controlled and the analysis of the performance required by the specific application. The dynamic model can provide a first guide for the selection of the control technique and can reveal possible limitations. Indeed, bear in mind that a control system is not only the control law. Computational resources, available electronics, and selection of sensors and actuators affect significantly the final performance. It has been shown that delays, sampling frequency, which can be seen as a sort of delay, presence of R.H.P. zeros, which is related to the position of sensors and actuators, and noise impose a limitation on the choice of the gains, and thus on the bandwidth. Moreover, the overall accuracy and repeatability is clearly affected by accuracy and repeatability of each element. Sometimes it is possible to select these components and try to optimize the design to achieve the specified performance. Other times, they are imposed by other constraints on the system or project. A model should be developed to describe the dynamics and include these effects. For the synthesis and update of the control, a control-oriented model could be more useful than an extremely detailed one (which should be used instead for validation) since it enables faster analyses and evaluations. In this model, only some features of the dynamics, which mostly affect the control performance, are modeled, and this selection must be done carefully. To understand better this point, an example is given. Consider the flexibility of a system, e.g., appendages or solar panels in a satellite. The designer could decide to neglect it in the model and synthetize a controller with a certain large bandwidth guaranteeing very good performance. When tested on a more realistic simulation environment, it could turn out that the closed-loop behavior is actually unstable. This is because flexibility imposes a limitation on the achievable control bandwidth. Should therefore the designer model as many flexible modes as possible? Not necessary, probably the main limitation is due to the first resonant frequency and modeling it would provide sufficient information to improve the control design. Control-oriented

model should capture all the essential dynamic features which impose some fundamental constraints to the control.

As mentioned previously in the chapter, stability of the closed-loop system is the first objective of the controller. To this aim, different methods have been reviewed, such as Nyquist criterion or the analysis of the poles/eigenvalues for linear system and Lyapunov and LaSalle theorems for nonlinear ones. When checking the stability, keep in mind that the response of the system to each input shall be verified, and not only to the setpoint. Indeed, in section [Chapter 10](#) – Control – Control design – Control design in frequency domain, it has been shown that even if the transfer function between the setpoint and the controlled variable could be stable, the other functions among the *Gang of four* may be not. For example, this could happen if an unstable pole is canceled by the controller. The complementary sensitivity function might be stable, but the unstable pole would remain in one of the other functions. From this observation, the following recommendation is recalled: cancellation of unstable poles should be avoided, and stability of all the transfer functions should be checked (internal stability), or, in general, the response to each identified input of the system.

Realistic simulation environment is often used to validate the controller before implementing it on the hardware. However, even if a very realistic simulator is available, uncertainties and unmodeled dynamics are always present in the real system. Therefore, the design of the controller should be developed in such a way to guarantee a certain robustness. Some stability margins have been introduced for this purpose (see [Chapter 10](#) – Control – Control design – Control design in frequency domain – Stability and stability margins), and reasonable values for them are recalled hereafter:

- Phase margin: $\phi_m \geq 30^\circ$.
- Gain margin: $g_m \geq 2$ (in decibel, ≥ 6 dB).
- Modulus margin: $s_m \geq 0.5$ (in decibel, ≥ -6 dB, or, equivalently, peak of the sensitivity function $M_S \leq 6$ dB).

When designing the control in the frequency domain, especially using loop-shaping technique or PI/PD/PID, the zeros of the controller are usually placed half a decade or a decade before the selected gain crossover frequency. Indeed, L.H.P. zeros provide an increase of the phase which can be beneficial to improve the gain margin.

Along with stability and robustness, some applications may require very challenging steady-state accuracy. Adding an integrator in the control law

can help meet this requirement. Indeed, integral term provides good steady-state tracking performance and rejection of constant disturbance. Clearly, the introduction of this action must be handled properly. Stability could be negatively affected, and antiwindup techniques shall be implemented to avoid the build-up of the error due to actuators' saturation.

Once designed the feedback control, it may happen that the performance of closed-loop system is still not satisfactory, especially in terms of tracking. To improve the system response, the designer could try to add a feedforward action, on the setpoint or/and on the disturbance, if measurable. It is recalled that this solution relies on the knowledge of the model, and thus a good identification of the parameters is recommended to increase its effectiveness.

As last remarks, it is important to always remember the trade-off nature of the control design. Indeed, designing a control system is basically seeking for a good balance between conflicting goals, taking into consideration the intrinsic limitations/constraints of the system.

References

- [1] K.J. Åström, R.M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*, second ed., Princeton University Press, 2021.
- [2] S. Skogestad, I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, Wiley, New York, 2007.
- [3] ECSS-E-ST-60-10C, “Space Engineering – Control Performance”, Noordwijk, 2008.
- [4] ECSS-E-HB-60-10A, “Space Engineering – Control Performance Guidelines”, Noordwijk, 2010.
- [5] W.J. Rugh, *Linear System Theory*, Prentice Hall, 1996.
- [6] T. Kailath, *Linear Systems*, vol 156, Prentice Hall, Englewood Cliffs, NJ, 1980.
- [7] K.J. Åström, Limitations on control system performance, *European Journal of Control* 6 (1) (2000) 2–20.
- [8] H.K. Khalil, *Nonlinear Systems*, third ed., Patience Hall, 2002.
- [9] J.-J.E. Slotine, W. Li, *Applied Nonlinear Control*, vol 199, Prentice Hall, Englewood Cliffs, NJ, 1991. No. 1.
- [10] W.J. Rugh, J.S. Shamma, Research on gain scheduling, *Automatica* 36 (10) (2000) 1401–1425.
- [11] M.J. Sidi, *Spacecraft Dynamics and Control: A Practical Engineering Approach*, vol 7, Cambridge University Press, 1997.
- [12] O.-E. Fjellstad, T.I. Fossen, Quaternion Feedback Regulation of Underwater Vehicles, 3rd IEEE Conference on Control Application, 1994.
- [13] J.G. Ziegler, N.B. Nichols, et al., Optimum settings for automatic controllers, *Transactions of the American Society of Mechanical Engineers* 64 (11) (1942) 759–768.
- [14] S. Silvestrini, J. Prinetto, G. Zanotti, M. Lavagna, Design of robust passively safe relative trajectories for uncooperative debris imaging in preparation to removal, *AAS/AIAA Astrodynamics Specialist Conference* (2020) 1–18.
- [15] A. Bemporad, M. Morari, Robust model predictive control: a survey, in: A. Garulli, A. Tesi (Eds.), *Robustness in Identification and Control. Lecture Notes in Control*

- and Information Sciences, vol 245, Springer, London, 1999, <https://doi.org/10.1007/BFb0109870>.
- [16] European Space Agency, “ECSS-E-HB-60-10A – control performance guidelines”, Technical Report, 2010.
 - [17] European Space Agency, “ESA pointing error engineering handbook, Handbook ESSB-HB-E-003”, Technical Report, 2011.
 - [18] M.E. Pittelkau, Pointing error definitions, metrics, and algorithms, *Advances in the Astronautical Sciences* 116 (2003) 901–920.
 - [19] T. Ott, W. Fichter, S. Bennani, S. Winkler, Precision pointing H_∞ control design for absolute, window-, and stability-time errors, *CEAS Space Journal* 4 (2013) 13–30, <https://doi.org/10.1007/s12567-012-0028-z>.
 - [20] K. Zhou, J. Doyle, K. Glover, Robust and Optimal Control, Prentice Hall, Upper Saddle River, NJ, 1996.
 - [21] C. Roos, Systems Modeling, Analysis and Control (SMAC) toolbox: an insight into the robustness analysis library, in: Proceedings of the IEEE Multiconference on Systems and Control, Hyderabad, India, August 2013, pp. 176–181.



FDIR development approaches in space systems

Massimo Tipaldi¹, Stefano Silvestrini², Vincenzo Pesce³,
Andrea Colagrossi²

¹University of Sannio, Benevento, Italy

²Politecnico di Milano, Milan, Italy

³Airbus D&S Advanced Studies, Toulouse, France

This chapter presents technical solutions and industrial processes used by the Space Industry to design, develop, test, and operate *health (or failure) management systems*, which are needed to devise and implement space missions with the required levels of dependability and safety (the former concerned with the ability of the spacecraft to deliver the intended service, while the latter more focused on functions that, if lost or degraded, can result into space mission loss) [1,2].

As shown later in this chapter, the design of health management systems needs the identification of potential risks that can threaten the main mission objectives, the assessment of their impacts on the spacecraft functions and services (both qualitative as well as quantitative), and the definition of mitigation means to prevent these risks from occurring. Space missions usually have to perform their tasks remotely under extreme environmental conditions and deliver high-quality services, sometimes for decades without interruption. On-board units and equipment operate in such extreme conditions with cutting-edge technologies. It is intuitively clear that dependability and safety are important design drivers for most types of space missions, as degradation in the quality of services is typically regarded as unacceptable.

Health management systems act as supervisory entities and ensure that the mission objectives and constraints are met, and that the spacecraft is protected from failures leading to a service deterioration or even worst to the spacecraft loss. The survival of the spacecraft has generally priority over its service availability. Health management systems monitor and process streams of observations in order to detect, isolate, and, if necessary, react to events and anomalies occurring inside the spacecraft.

In the Space Industry, health (or failure) management is commonly referred to as *Failure (or Fault) Detection, Isolation and Recovery* (FDIR, note

that “F” can stand for fault (state) or failure (event) indistinctively), and it is regarded as a complex system engineering discipline [1–3]. FDIR system development is addressed since the very beginning of any space mission design [4] and plays a relevant role in the definition of their reliability, availability, and safety objectives. If not correctly managed, the development and testing of FDIR capabilities can cause project schedule delays and cost overruns [5].

Space is actually a harsh and challenging environment, where it is absolutely necessary to provide spacecraft with adequate hardening. First, very stringent quality requirements for the design and testing processes of the spacecraft have to be met [6]. But this is not sufficient to address dependability and safety requirements. Thus, FDIR systems have to be engineered and developed in order to provide specific functionality, able to handle situations when the abovementioned risks can become reality.

In this chapter, both technical and programmatic FDIR strategies are presented along with their strong connection with the wider concept of on-board autonomy, which is becoming the key point in the design of new-generation spacecrafts. Today’s rapid progress in on-board computational power supports the transfer of FDIR functions from the ground to the flight segment and the enhancement of the spacecraft on-board autonomy [3,7]. FDIR system capabilities are usually interwoven with all other features of the system, and this is one of the main causes of FDIR system design complexity [1,2]. The European Space Agency (ESA) has recognized the system-level role of the FDIR. In this regard, in 2016, the SAVOIR-FDIR working group was established with the aim of harmonizing the vocabulary, the practices, and the expectations related to the design of the FDIR components within a system perspective, see Ref. [8].

The overall chapter is inspired by FDIR systems designed for ESA missions; however, the presentation is maintained at a proper level of detail so that its contents are in line with the FDIR practices adopted by other space agencies. FDIR is mentioned in several European Cooperation for Space Standardization (ECSS) documents. However, such ECSS standards do not provide guidelines on how to design, implement, and test FDIR systems and do not describe the interfaces with the other engineering disciplines (e.g., Reliability, Availability, Maintainability, and Safety, software [SW], and operations). Such information is spread in different ECSS documents, and it is difficult to have a comprehensive overview of FDIR system development processes. This chapter tries to solve such issue; for more details, readers are encouraged to refer to the *SAVOIR FDIR Handbook* [8].

The content of this chapter is structured as follows:

- *FDIR in space missions, terms, and definitions.* This section provides an overview on FDIR systems and the related terminology. It also includes an introduction to FDIR specifically targeted to space missions.
- *Current FDIR system development process and industrial practices.* In this section, the most relevant current industrial approaches and technical solutions used to design, develop, test, and operate FDIR systems are outlined.
- *FDIR system hierarchical architecture and operational concepts.* This section briefly explains FDIR system hierarchical architecture.
- *FDIR system implementation in European space missions.* The most relevant FDIR system implementations according to European standards are introduced in this section.
- *FDIR system verification and validation approach.* This section summarizes and highlights the most relevant FDIR system verification and validation approaches.
- *FDIR concept and functional architecture in GNC applications: a short overview.* In this section, the concept and functional architecture of FDIR techniques for spacecraft guidance, navigation, and control (GNC) applications are detailed.



FDIR in space missions, terms, and definitions

In current and upcoming space missions, the need of designing spacecraft with a high level of on-board autonomy is emerging [3,9]. This trend is particularly evident for spacecraft operating in challenging contexts, which feature deep-space exploration systems or critical operational phases, such as automated maneuvers for space rendezvous. From an operational point of view, on-board autonomy can be regarded as migration of functionality from the ground segment to the flight segment [10]. The implementation of on-board autonomy depends on the specific mission requirements and constraints and can therefore vary between a very low level of autonomy (which implies a relevant involvement of the ground segment in space mission operations) to a high level of autonomy (whereby, most of the functions are performed on-board [7]). As shown in Refs. [3,10] and in line with Ref. [7], on-board autonomy can be associated to the following application fields:

- *Intelligent sensing.* The ability to infer system state from the environmental sensor data.

- *Mission planning and execution.* The process of planning and executing both nominal and contingency mission operations satisfying temporal and resource constraints.
- *On-board FDIR.* The ability of on-board detecting, isolating, and recovering from failure situations.
- *Distributed decision-making.* Effective cooperation among independent autonomous spacecraft in order to achieve common goals.

Drivers for increased on-board autonomy are the improvement of spacecraft dependability [1,4,5], the need to overcome long communication delays and outages [11], and the reduction of costs in ground segment operations [12]. This way, important requirements can be fulfilled, such as continuous mission product generation on board, real-time spacecraft control outside ground contact, maximization of mission objectives in relation to the available on-board resources, and robust operations in the presence of on-board failures and context uncertainty.

The need of designing more and more autonomous and dependable missions is the main factor leading toward the enhancement of spacecraft FDIR systems and their related architecture. FDIR systems act as supervisory controllers, preventing the spacecraft behavior from the undesired scenarios. FDIR systems are basically featured by the following capabilities [1,2]:

- *Detection.* The determination of the presence of faults in a system and their times of occurrence.
- *Isolation.* The determination of their location, type, and severity.
- *Recovery.* The process of choosing the best action to recover from the anomaly condition (for instance, by selecting the most adequate spacecraft reconfiguration, which is determined by the remaining healthy components).

FDIR requirements have to be scaled and deployed between the flight and the ground segments according to some factors, such as mission type and objectives, spacecraft orbit and ground visibility profile, operations concepts, communication bandwidth, and latency [1–3]. Moreover, important constraints (e.g., robustness, reactive detection, quick isolation/identification, and limited on-board resources – central processing unit (CPU) and memory) on FDIR solutions have to be taken into account.

FDIR system conception and implementation in a space system can be a rather complex field, since the analysis of spacecraft failures can extend to very sophisticated considerations. The cause of malfunctions could be one or more failed components, incorrect control actions, or external disturbances, which impact system operations. In complex, heterogeneous systems

with large numbers of interacting physical components, the effects of faults can propagate through the boundaries of many subsystems and can result in diverse fault symptoms. FDIR information processing and physical components are tightly integrated and intertwined; therefore, the structure and properties of physical spacecraft components determine the functions to be implemented by the FDIR systems.

To give further insights into the FDIR complexity, the FDIR system design strongly depends on the chosen spacecraft operational concept [1,8]. In this regard, FDIR systems can be shaped between two extremes, i.e., having a straightforward on-board FDIR implementation (where only vital elements are checked on board, the related recovery is completely performed by the ground) or making use of a more sophisticated on-board FDIR design (where failures are identified at the lowest level possible and solved autonomously by the spacecraft and the ground intervention is reduced to the minimum). In making this choice, it is necessary to make a trade-off analysis by taking into account different criteria, such as, mission outage risks, on-board software (OBSW) development costs, hardware (HW) redundancy costs.

Indeed, FDIR systems have to be carefully designed in order to exploit recurrent and reusable approaches throughout the different missions. It is also necessary to assess how to implement their failure management and recovery capabilities, e.g., via HW or via the OBSW, the latter being the application SW running on the main spacecraft on-board computer (OBC). As better clarified later in this chapter, the OBSW plays a relevant role in the implementation of FDIR system capabilities. The decision as to whether FDIR functions are to be implemented, i.e., via HW or OBSW functions, is of prime importance: the advantage of the OBSW lies in the fact that OBSW offers in-flight modification capability [13].

Current FDIR systems perform system monitoring through the available on-board observations, diagnose off-nominal conditions, and initiate the recovery procedures, for which the purpose is to restore the spacecraft operation to nominal conditions (if feasible), to a degraded operation (if possible), or, as a last resort, to simply put the spacecraft into a known safe configuration [14,15] (where further investigations are carried out by the ground station). Possible recoveries are the retry operation, the unit reboot, and the switch to either a redundant unit or to a completely redundant string (in case of no failure identification). To avoid the loss of platform functions mandatory for the mission, the redundancy concept has to be such that the functionality for which the failure can lead to mission loss is to be

protected by independent redundant alternatives [1–3]. Hot redundancy can be adopted for such vital spacecraft functions.

Another interesting aspect in the development of FDIR systems is to understand which spacecraft subsystems and units are more likely to be affected by failures and the corresponding impact on the overall mission. This analysis is fundamental for the FDIR system functionality definition and can support project programmatic/system-level decisions, e.g., where to allocate the budget foreseen for the FDIR system development. In this regard, we can mention the study by Tafazoli [16], where on-orbit spacecraft failures from 1980 to 2005 were reviewed. As reported in Ref. [16], most of the failures affected the Altitude and Orbit Control (AOCS) and the Power subsystem, in particular: AOCS (32%), Power (27%), Command and Data Handling (15%), Telemetry (TM), Tracking and Command (12%), and other subsystems (14%).

Terms and definitions

Finally, a few relevant terms used in the FDIR realm are hereafter explained (an exhaustive list can be found in Ref. [8]):

- *Anomaly*. Any deviation from the expected situation.
- *Criticality*. Combined measure of the severity of a failure mode and its probability of occurrence.
- *Dependability*. The extent to which the fulfillment of a required function can be justifiably trusted.
- *Failure*. The event resulting in an item being no longer able to perform its required function.
- *Failure cause*. The presumed cause associated to a given failure mode.
- *Failure effect*. Consequence of an assumed item failure mode on the operation, function, or status of the item.
- *Failure isolation*. Functionality to isolate the failed unit or subsystem, to avoid failure propagation and deterioration of the impacted equipment.
- *Failure mode*. Mechanism through which a failure occurs.
- *Failure mode, effects, and criticality analysis (FMECA)*. Analysis by which each potential failure mode in a product (or function or process) is analyzed to determine its effects, and then classified according to its criticality.
- *Fault*. State of an item characterized by inability to perform as required.
- *Functional chain*. Set of HW and/or SW units operating together to achieve a given subset of closely related functions of a system.

- *On-board autonomy.* Capability of the space segment to manage nominal or contingency operations without ground segment intervention for a given period of time.
- *Redundancy.* Existence of more than one means for performing a given function with the intention of increasing dependability.
- *Safe mode.* Spacecraft operating mode guaranteeing the spacecraft safety that can be autonomously sustained for an indefinite period of time (or at least for a period of time significantly longer than the ground reaction time) and in which basic spacecraft functions, including permanent communications with ground, are available to the operators to perform the recovery procedures.
- *Safety.* State where an acceptable level of risk is not exceeded (risks related to, e.g., fatality, damage to the main functions of a flight system itself, damage to launcher HW or launch site facilities).
- *Service.* Functional element of the space system that provides a number of closely related functions that can be remotely operated.

Most of the definitions are taken from the ECSS standards, for instance, see Refs. [7,17,18] and Ref. [19].



Current FDIR system development process and industrial practices

This paragraph outlines the current industrial approaches and practices used to conceive, design, implement, test, and operate FDIR systems. Fig. 11.1 shows the different phases featuring the FDIR system development and their relationship with the spacecraft project phases. The following major steps are identified [20]:

- *Study phase.* It includes the identification of autonomy needs, system-level requirements, and ground segment constraints. This step also includes the definition of the preliminary FDIR concept on a mission level.
- *Definition phase and the early development phase.* It includes the definition of the FDIR system for the required mission (e.g., identification of failures on system/subsystem/unit levels), the definition of the overall FDIR system architecture (FDIR hierarchy and dependencies), the definition of the FDIR functions, and their mapping versus HW or/and SW elements.
- *Implementation and testing phase.* It includes the FDIR system detailed design, implementation, and testing.
- *In-orbit phase.* It includes the FDIR system maintenance and operations.

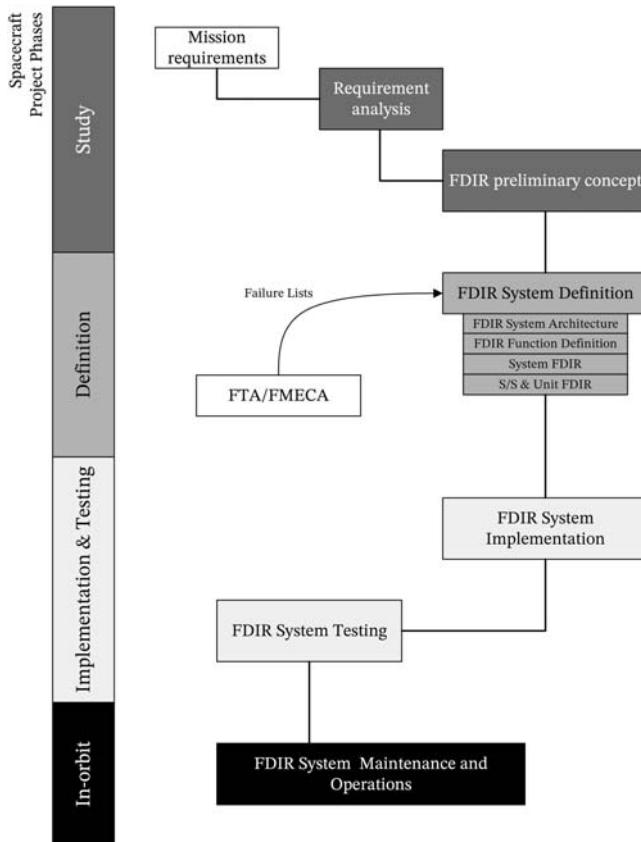


Figure 11.1 FDIR system development versus spacecraft project phases.

It is evident that FDIR system discipline has to be regarded as a broad system-level activity and requires in-depth knowledge of space system engineering. The study and the definition phases are fundamental for the development of FDIR systems from both a technical and programmatic standpoint. As shown in Fig. 11.1, FDIR system requirements and functions stem from the mission/high-level system needs and constraints and the outcome of the Fault Tree Analysis (FTA) and FMECA. FTA and FMECA results are often combined for the FDIR system functionality definition. FTA is a deductive analysis where top-level failures (e.g., loss of power or loss of fuel) are addressed and all the basic faults that can lead to such states are identified. This process results in a tree-shaped diagram. The identified failure scenarios need to be managed by the FDIR system. FMECA is an inductive analysis, and it computes the causal effects on the system of a given

fault combination. Examples of fault-tree diagrams can be found in Ref. [6]. For any identified relevant failure modes in the FMECA, which are not already covered by the FDIR functions derived from the FTA, additional FDIR functions are defined (bottom-up approach). After specifying all the FDIR functions, the implementation into the OBSW and/or HW units is performed, followed by a testing of all the implemented FDIR requirements. During the operational phase, the FDIR system capabilities can be tuned and even enhanced, thanks to the lessons learnt from the spacecraft operations in its final environment and the availability of flight data [21,22].

FDIR system hierarchical architecture and operational concepts

Current European space missions usually implement the FDIR systems via a hierarchical architecture [8,20,23]. A typical example of such layered architecture is shown in Fig. 11.2 and consists of the following layers:

- *Level 0.* Unit failure autonomously detected and recovered within the unit itself and reported to the supervision level (which is usually the OBSW). Examples of level 0 failures are short currents, overvoltage, and data bus failures.

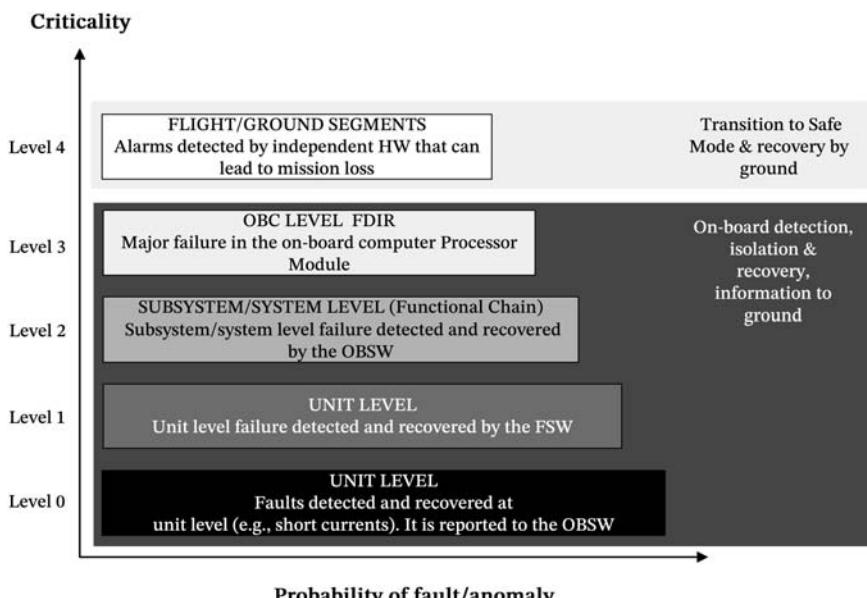


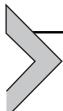
Figure 11.2 FDIR system hierarchical structure.

- *Level 1.* Unit failure detected and recovered by the supervising OBSW, which can command, e.g., the unit reset. They can be recovered by resorting to the redundant path. As for the fault detection, the OBSW has to monitor unit data, e.g., against specific thresholds.
- *Level 2.* Functional chain (i.e., subsystem/system) level failure detected via anomalous performance parameters by the OBSW. The recovery action is also performed by the OBSW. Level 2 failures are usually caused by the propagation of undetected/unsolved unit failures. Subsystem/system-level data checks and functional monitoring characterize this level. Examples of Level 2 failures are the spacecraft orientation outside the operational range and its anomalous angular rate.
- *Level 3.* Major failure in the OBC Processor Module, detected either by the OBSW or by a supervising equipment (e.g., the Reconfiguration Module [13,15]). Examples of Level 3 failures are: processor undervoltage, starvation/crash of the OBSW detected by missed refresh of the watchdog, invalid instruction, or memory access.
- *Level 4.* Major spacecraft failure detected at HW level by the Reconfiguration Module. Examples are: major battery discharge, sun intrusion through the optical payload field of view, and excessively long thruster firing.

The abovementioned FDIR system design consists in allocating FDIR functionality into the levels of the FDIR hierarchical architecture. As a matter of fact, failures are usually deployed along five hierarchical levels and are characterized by a specific severity, the function(s) involved in their detection, and the recovery sequence. The highest FDIR level is usually in charge of the correct execution of vital functions of the spacecraft, whereas lower-level hierarchies operate at unit level. A higher level is triggered by the adjacent lower level, only when the latter is not able to solve/isolate a fault. In such a case, FDIR functions allocated into the higher level can perform the related recovery action by exploiting functions implemented by the next lower level.

Most of the FDIR functions are implemented by the OBSW [1–3]. The operational concept of a typical spacecraft includes one or more safe mode configurations. They can represent the ultimate reaction to spacecraft severe anomalies. Indeed, safe modes can be entered when the on-board FDIR system is not able to solve critical on-board failure conditions (such as a severe spacecraft attitude excursion outside the operational range) or by means of specific telecommands (TCs) sent from the ground segment. The spacecraft can remain in this mode without ground segment intervention for a specified period of time. During the safe mode, the communication link to the

ground segment, a specific power supply profile, and thermal survival functions for relevant equipment are maintained, whereas all nonessential on-board functions are powered off or disabled. The recovery of the spacecraft from safe mode to nominal mode needs to be commanded by ground. FDIR Level 3 anomalies are usually related to the spacecraft OBC, which executes the central OBSW. At this level, failure management consists in executing an OBC reset, specific reconfiguration procedures, and a transition into safe mode as appropriate.



FDIR system implementation in European Space missions

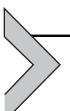
FDIR functions deployed into Level 1 and Level 2 are completely handled by the OBSW. As highlighted in Refs. [1,2,8], FDIR systems in European Space projects are usually implemented via the Packet Utilization Standard (PUS) services (ECSS-E-ST-70-41C—TM and TC packet utilization [19]). Such standard addresses the role of both the flight and ground segments in implementing FDIR systems.

Generally speaking, the PUS defines the operational model of a spacecraft, which consists of a set of extensible services that can be invoked through a service request (TC source packet), with such an invocation resulting in the generation of zero or more service reports (TM source packets). Among the other things, the PUS addresses the FDIR operational concepts and provides a set of services that allows the standardization of the failure management between ESA missions. In particular, the following services concern the FDIR functionality: the On-board Monitoring PUS Service 12, the On-board Event Reporting PUS Service 5, and the On-board Event—Action PUS Service 19. The latter triggers some on-board actions that can be a single TC, the execution of a command sequence (PUS Service 21), or the execution of an On-Board Command Procedure (OBCP, PUS Service 18 [24]). Avoidance of multiple recovery execution for simultaneous occurrence of multiple related failures is achieved by the combination of PUS service 12 parameter monitoring items into a single “functional monitoring” which issues a dedicated event report triggering the single recovery.

PUS-based FDIR systems are industrially mastered and established within the current industrial spacecraft development processes [2,19]. Moreover, being based on configuration tables (and thus, modifiable at run-time via TC source packets without the need of patching the whole OBSW), the PUS services can offer some flexibility to the overall FDIR system during

both the ground testing and the spacecraft operational life. For instance, the monitored item thresholds and the list of on-board actions to be triggered (in case of threshold violations) are implemented via configuration tables and can easily be updated during the space mission operational phase.

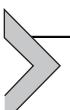
The PUS standard can be also tailored in order to accommodate mission-specific requirements [3,19]. However, the overall FDIR concept is based on a threshold monitoring mechanism, and this can imply some limitations in the diagnostics capabilities of the resulting FDIR system. Indeed, threshold-based diagnostic routines process symptoms in isolation, which may result in incorrect diagnoses or contradictory deductions. Limitations of current FDIR solutions adopted in Space Industry are addressed in Ref. [25], where model-based FDIR solutions are explored.



FDIR system verification and validation approach

Before being operated, FDIR systems need to be verified and validated on-ground by using simulators and different spacecraft engineering models [1,2,8,26]. Traditional system-level testing basically aims at confirming that each command works as expected, each functional and performance requirement has been met, and that all sequences of commands that will probably be used during the mission work. Each requirement must have a corresponding verification requirement, which dictates the test, analysis, review of design, or inspection procedure that will be used to verify the corresponding requirement. Such approaches have also been proved to be quite effective for the FDIR systems so far [27]. Most of the FDIR systems functions are implemented by the OBSW; therefore, FDIR system verification and validation activities are intertwined with the ones concerning the OBSW.

OBSW requirements derived from FDIR requirements and corresponding SW artifacts normally undergo a well-defined OBSW verification and validation process, see Ref. [28]. Once all the HW and SW items concurring at the implementation of system-level FDIR requirements have been verified and integrated, FDIR requirements are globally verified at system level [8,26,28].



FDIR concept and functional architecture in GNC applications: a short overview

A large body of literature and industrial applications of fault-tolerant GNC solutions is available. Such solutions have been applied to different

contests, such as precise formation flying [29,30], small body descent and landing missions [31], deep space missions [32], and reentry modules [33]. Depending on mission and system-level requirements, different FDIR approaches have been conceived, designed, and implemented. Generally speaking, the main objectives of a fault-tolerant GNC are [34]:

- Early detection and diagnosis of anomalies, i.e., detection delay should be minimized.
- Good ability to discriminate between different failures (isolation).
- Good robustness to various noise and uncertainties sources, and their propagation through the system.
- High sensitivity and performance, i.e., high detection rate and low false alarm rate.

In most GNC systems endowed with FDIR capabilities, failures and recoveries are allocated into the levels of the FDIR hierarchical architecture as shown in Fig. 11.2 [30,32,35]. Moreover, specific FDIR modes are designed and incorporated in the GNC system to handle critical failure conditions. An interesting case is the PROBA 3 mission [30], which is devoted to the demonstration of precise formation flying technology: the safe mode is associated to safe trajectories (where no formation orbit control for long periods of time is required), while the collision avoidance mode corresponds to drifting orbits (designed with the objective of avoiding threatening collisions). More precisely, the safe mode is close to the classical safe mode of a single satellite and is designed to ensure no collision with the other satellites of the formation, whereas CAM is a fail-operational mode, wherein the spacecraft attempts to recover autonomously its operational conditions [35].

In fault-tolerant GNC systems, autonomous failure management capabilities are intertwined with on-board mission planning and execution [3,33,36]. From an architectural standpoint, this means that fault-tolerant GNC systems include three main functional modules [33,36,37]: the Mission and Vehicle (or Spacecraft) Manager (MVM), the GNC Manager, and the GNC module itself (see Fig. 11.3). In particular:

- The MVM is responsible for the accomplishment of the mission goals by programming the vehicle activity plans and monitoring their executions. It defines the sequences of phases, modes, and maneuvers along with the scheduling of equipment for such modes. It also implements the TC/TM interface with the ground segment.
- The GNC Manager receives the commands from the MVM, manages the execution of the corresponding GNC functions/modes, and reports their execution to the MVM. Such module also includes the event-

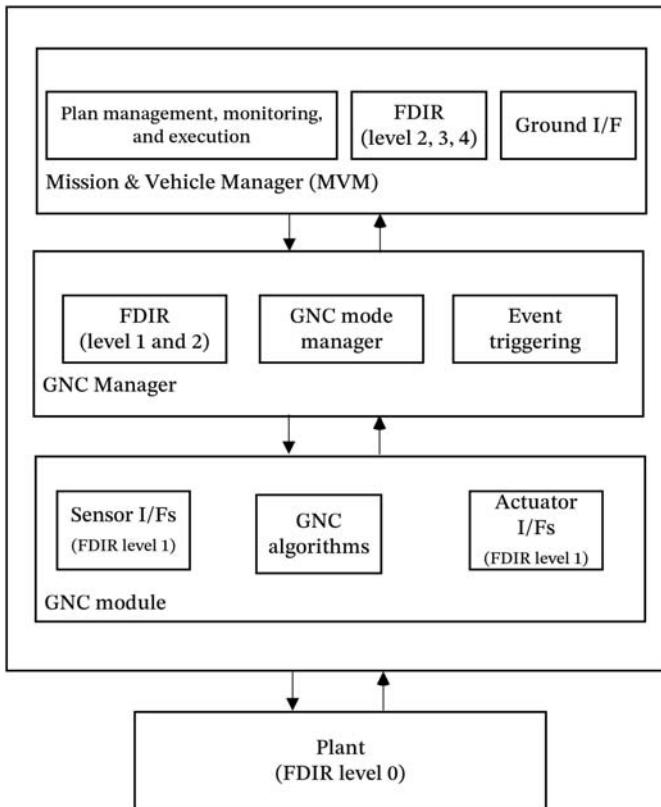


Figure 11.3 Fault-tolerant GNC functional architecture.

triggering function, which is in charge of generating relevant mission-level events (e.g., parachute deployment).

- The GNC module provides all the GNC algorithms needed in the different phases. Moreover, it implements the functional interfaces toward the sensors (e.g., camera, GPS receiver, gyroscope) and actuators (e.g., thruster, reaction wheel, magneto-torquer). Based on the current mode, the corresponding algorithms, sensors, and actuators are enabled.

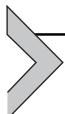
The overall system is in charge of the execution of the various GNC modes, the switching between modes, the assignment of resources to each mode, the detection of failures and reconfiguration after failures, the high-level monitoring of GNC functions and of the vehicle state. FDIR capabilities can be allocated throughout the abovementioned blocks (see Fig. 11.3, only as an example). Unit-level FDIR capabilities can be allocated

into the GNC module itself, while mission-level FDIR can be provided by the MVM module. In this regard, if needed, mission objectives can change as a result of a mission-level recovery action, e.g., the reshape of a new reentry trajectory for a high or medium lift-to-drag ratio vehicle as reported by Ref. [34].

References

- [1] M. Tipaldi, B. Bruenjes, Survey on fault detection, isolation, and recovery strategies in the space domain, *Journal of Aerospace Information Systems* 12 (2) (2015) 235–256.
- [2] X. Olive, FDI (R) for satellites: how to deal with high availability and robustness in the space domain? *International Journal of Applied Mathematics and Computer Science* 22 (2012) 99–107.
- [3] M. Tipaldi, L. Glielmo, A survey on model-based mission planning and execution for autonomous spacecraft, *IEEE Systems Journal* 12 (4) (2017) 3893–3905.
- [4] M. Tipaldi, B. Bruenjes, Spacecraft health monitoring and management systems, in: *IEEE Metrology for Aerospace (MetroAeroSpace)*, 2014, pp. 68–72, 2014.
- [5] L.M. Fesq, Current fault management trends in NASA’s planetary spacecraft, in: *IEEE Aerospace Conference*, 2009, pp. 1–9, 2009.
- [6] Space mission analysis and design, in: J.R. Wertz, W.J. Larson (Eds.), *3rd Space Technology Series*, third ed., Microcosm Press and Kluwer Academic Publishers (Jointly), Boston MA, 1999.
- [7] ECSS-E-ST-70-11C space engineering—space segment operability, European Cooperation for Space Standardization Standard, 2008.
- [8] SAVOIR-FDIR Handbook, European Space Agency, 2019. <https://essr.esa.int>.
- [9] A. Jónsson, R.A. Morris, L. Pedersen, Autonomy in space: current capabilities and future challenge, *AI Magazine* 28 (4) (2007), 27–27.
- [10] T. Grant, A.O. Soler, A. Bos, U. Brauer, M. Neerinex, M. Wolff, Space autonomy as migration of functionality: the Mars case, in: *2nd IEEE International Conference on Space Mission Challenges for Information Technology, SMC-IT’06*, 2006, pp. 1–7.
- [11] R. Sterritt, C.A. Rouff, M.G. Hinckey, J.L. Rash, W. Truszkowski, Next generation system and software architectures: challenges from future NASA exploration missions, *Science of Computer Programming* 61 (1) (2006) 48–57.
- [12] J. van der Ha, Trends in cost-effective mission operations, *Acta Astronautica* 52 (2–6) (2003) 337–342.
- [13] M. Tipaldi, C. Legendre, O. Koopmann, M. Ferraguto, R. Wenker, G. D’Angelo, Development strategies for the satellite flight software on-board Meteosat Third Generation, *Acta Astronautica* 145 (2018) 482–491.
- [14] V. Nardone, A. Santone, M. Tipaldi, D. Liuzza, L. Glielmo, Model checking techniques applied to satellite operational mode management, *IEEE Systems Journal* 13 (1) (2018) 1018–1029.
- [15] M. Tipaldi, M. Witzmann, M. Ferraguto, L. Glielmo, An approach for geostationary satellite mode management, *IFAC-PapersOnLine* 50 (1) (2017) 7241–7246.
- [16] M. Tafazoli, A study of on-orbit spacecraft failures, *Acta Astronautica* 64 (2–3) (2009) 195–205.
- [17] ECSS-S-ST-00-01C – Glossary of Terms, European Cooperation for Space Standardization Standard, 2012.
- [18] ECSS-Q-ST-30-02C – Failure Modes, Effects (And Criticality) Analysis (FMEA/FMECA), European Cooperation for Space Standardization Standard, 2009.
- [19] ECSS-E-ST-70-41C – Telemetry and Telecommand Packet Utilization, European Cooperation for Space Standardization Standard, 2016.

- [20] R. Gessner, B. Kosters, A. Hefler, R. Eilenberger, J. Hartmann, M. Schmidt, Hierarchical FDIR concepts in S/C systems, in: Space OPS 2004 Conference, 2004, pp. 1–11, 2009.
- [21] M. Tipaldi, L. Feruglio, P. Denis, G. D’Angelo, On applying AI-driven flight data analysis for operational spacecraft model-based diagnostics, *Annual Reviews in Control* 49 (2020) 197–211.
- [22] J.A. Martínez-Heras, A. Donati, Enhanced telemetry monitoring with novelty detection, *AI Magazine* 35 (4) (2019) 37–46.
- [23] A. Wander, R. Förstner, Innovative Fault Detection, Isolation and Recovery Strategies On-Board Spacecraft: State of the Art and Research Challenges, Deutsche Gesellschaft für Luft-und Raumfahrt-Lilienthal-Oberth eV, 2013.
- [24] M. Tipaldi, M. Ferraguto, T. Ogando, G. Camatto, T. Wittrock, B. Bruenjes, L. Gielmo, Spacecraft autonomy and reliability in MTG satellite via On-board Control Procedures, in: 2015 IEEE Metrology for Aerospace (MetroAeroSpace), 2015, pp. 155–159.
- [25] J. Marzat, H. Piet-Lahanier, F. Damongeot, E. Walter, Model-based fault diagnosis for aerospace systems: a survey, *Journal of Aerospace Engineering* 226 (10) (2012) 1329–1360.
- [26] M. Zoppi, M. Tipaldi, A. Di Cerbo, Cross-model verification of the electrical power subsystem in space projects, *Measurement* 122 (2018) 473–483.
- [27] K. Reinholz, K. Patel, Testing autonomous systems for deep space exploration, *IEEE Aerospace and Electronic Systems Magazine* 23 (9) (2008) 22–27.
- [28] ECSS-E-ST-40C – Software, European Cooperation for Space Standardization Standard, 2009.
- [29] G. Di Mauro, M. Lawn, R. Bevilacqua, Survey on guidance navigation and control requirements for spacecraft formation-flying missions, *Journal of Guidance, Control, and Dynamics* 41 (3) (2018) 581–602.
- [30] J.S. Llorente, A. Agenjo, C. Carrascosa, C. de Negueruela, A. Mestreau-Garreau, A. Cropp, A. Santovincenzo, PROBA-3: precise formation flying demonstration mission, *Acta Astronautica* 82 (1) (2013) 38–46.
- [31] D. Ge, P. Cui, S. Zhu, Recent development of autonomous GNC technologies for small celestial body descent and landing, *Progress in Aerospace Sciences* 110 (2019).
- [32] A. Zolghadri, D. Henry, J. Cieslak, D. Efimov, P. Goupil, *Fault Diagnosis and Fault-Tolerant Control and Guidance for Aerospace Vehicles*, Springer, London, UK, 2014.
- [33] F. Cacciatore, et al., The design of the GNC of the Re-entry module of space rider, in: European Conference for Aeronautics and Space Sciences, EUCASS, 2019.
- [34] A. Zolghadri, Advanced model-based FDIR techniques for aerospace systems: today challenges and opportunities, *Progress in Aerospace Sciences* 53 (2012) 18–29.
- [35] L. Pirson, J. Christy, B. Udrea, ICC2 study: GNC development in formation flying, *IFAC Proceedings Volumes* 40 (7) (2007) 828–833.
- [36] M. Jankovic, J. Paul, F. Kirchner, GNC architecture for autonomous robotic capture of a non-cooperative target: preliminary concept design, *Advances in Space Research* 57 (8) (2016) 1715–1736.
- [37] W.H. Fehse, Control tasks in automatic rendezvous and docking of spacecraft, in: 1999 European Control Conference, ECC, 1999, pp. 4842–4848.



GNC verification and validation

Francesco Pace¹, Emanuele Paolini², Francesco Sanfedino³,
Daniel Alazard³, Andrea Colagrossi⁴, Vincenzo Pesce⁵,
Stefano Silvestrini⁴

¹GMV Aerospace & Defence, Madrid, Spain

²D-Orbit, Fino Mornasco, Italy

³ISAE-SUPAERO, Toulouse, France

⁴Politecnico di Milano, Milan, Italy

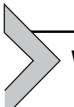
⁵Airbus D&S Advanced Studies, Toulouse, France

This chapter presents an overview of the key concepts regarding guidance, navigation, and control (GNC) verification and validation procedures. Model-in-the-loop (MIL), software-in-the-loop (SIL), processor-in-the-loop (PIL), and hardware-in-the-loop (HIL) tests are presented from GNC modeling and simulator development. Autocoding and different verification activities, including requirements, code standards, and coverage verification are discussed. Finally, in-orbit testing and all the activities aiming to verify correct high-level, in-orbit functionalities, performances, and operations of the system are introduced.

The content of this chapter is structured as follows:

- *Why it is important?* This section introduces the main concepts and definitions of GNC verification and validation (V&V) procedures. Their importance in the GNC design and implementation phase is highlighted.
- *Statistical methods.* In this section, the most relevant statistical methods to conduct successful and representative GNC V&V campaigns are outlined.
- *MIL test.* This section introduces the concept of MIL test. High-level guidelines for GNC algorithms modeling, including architectures and implementation rules, are described. MIL verification activities are then detailed for this test phase.
- *SIL/PIL test.* Similarly, SIL and PIL tests are introduced. Standard guidelines for autocoding are presented along with verification activities at SIL and PIL level.

- *HIL test.* This section summarizes and highlights the most relevant steps to be followed during HIL test and its main objectives. Examples of hardware and software HIL verification are discussed.
- *In-orbit test.* In this section, the concept of in-orbit testing for system functionalities, performance, and operations verification is detailed.



Why it is important?

As in all industrial process, the GNC system after a preliminary design phase needs a consolidation through a V&V phase.

This phase is one the most important in a spacecraft design cycle since it provides the certificate that the product will finally work as specified by its requirement list and will assure the accomplishment of the mission by providing the same degradation of performance in case of predicted anomalies.

Following the definitions provided by the glossary of terms of the European Cooperation for Space Standardization (ECSS) [1]:

- “(A) **verification process** [...] demonstrates through the provision of objective evidence that the product is designed and produced according to its specifications and the agreed deviations and waivers, and is free of defects. A waiver can arise as an output of the verification process. Verification can be accomplished by one or more of the following methods: analysis (including similarity), test, inspection, review of design.”
- “(A) **validation process** [...] demonstrates that the product is able to accomplish its intended use in the intended operational environment. Verification is a prerequisite for validation.”

Note that a “*waiver* (is a) formal authorization to accept products which during production, or after having been submitted to inspection or tests, are found to depart from specified requirements. Deviation is an *a priori* decision whereas waiver is an *a posteriori* decision with respect to the production phase.”

Validation is achieved when the product meets the needs linked to its use (i.e., level of pointing performance requested by the client). If these needs are systematically translated into requirements in the verification process, their achievement automatically brings the validation process to its fulfillment.

The “Satellite attitude and orbit control system (AOCS) requirements” document provided by the ECSS [2] provided some guidelines on how to standardize the GNC requirements by covering the following areas:

- Attitude estimation.
- Attitude guidance.
- Attitude control.
- Orbit control.
- Orbit estimation (or navigation).
- Acquisition and maintenance of a safe attitude in emergency cases and return to nominal mission upon command.

For missions where pointing performance constraints spacecraft design, as for Earth observation or Science, the ECSS formalizes a list of pointing requirements [3,4], which have been detailed in [Chapter 10 - Control](#).

For the GNC system, V&V is an incremental process applied to each of the following GNC development/realization steps:

- MIL simulation.
- SIL simulation.
- PIL simulation.
- HIL simulation.
- In-orbit test (IOT).

The principle of these steps is to progressively move from the system modeling environment to the real-world implementation. The latter one has, in fact, to deal with software limits due to the translation into real-time code, processor maximum achievable performance, unavoidable mismatch between plant/sensors/actuators models and real hardware, and in-orbit environment as well.

One of the most widespread tools used for GNC prototyping and simulation is MATLAB/Simulink where multiphysics nonlinear simulators can be developed to validate guidance and control laws.

General practice is to start with very simple models at the beginning of a project, which capture the main dynamics for the preliminary dimensioning of the sensors/actuators according to the maximum expected control/torques. In this spirit, if the linearity hypothesis is justified, as in the majority of spacecraft missions where the maximum spacecraft speed rates are very small, linear models or a family of linear models (for different mission control phases) can be derived with the advantage of disposing of a huge amount of control techniques for which stability and performance robustness can be analytically proven in frequency domain by avoiding time-consuming time simulations. For instance, as seen in [Chapter 10 - Control](#), different metrics are, in fact, available to validate stability of a control design: from the classical ones as gain, phase, delay, and modulus margin to more recent as the structured singular values theory [5].

For spacecraft with large and flexible structures (like big antennas), a rigid body approximation is not sufficient to correctly predict and guarantee the stability and performance of the real system. It is then important to validate the GNC models with accurate finite element structural models.

Once preliminary control laws are verified on linear-based plants, a validation in a high-fidelity nonlinear simulator is always needed. In this case, finer models are used to simulate not only the plant, for which nonlinear terms (as Coriolis or centrifugal forces) are not neglected this time but also the sensor/actuator dynamics and the environmental conditions. In this phase, for instance, actuator/sensor strong nonlinearities (such as saturations, dead-zones, and hysteresis) are taken into account together with high-fidelity models of orbital and attitude perturbations (gravity gradient torque, solar pressure, atmospheric drag, etc.).

If the plant linear hypothesis is confirmed and the control design avoids strong nonlinearities, like saturations by limiting, for example, the maximum control signal before entering into the nonlinear behavior, the linear-based model can be sufficient to validate the design. On the other hand, if stability and/or performance drop to unacceptable level, a control redesign is needed taking into account neglected dynamics or unpredicted behaviors. If nonlinearities cannot be absolutely neglected, then, a nonlinear controller has to be synthesized as shown, for instance, in [Chapter 10](#) – Control.

The widespread way to validate a nonlinear model simulator in industry is to use Monte Carlo simulations where a probability distribution function (PDF) is associated to key parameters for which a fixed level of misknowledge (i.e., manufacturing imperfections for mechanical parameters) or statistical nature (i.e., actuator/sensor noises and perturbation) is defined. The counterpart of Monte Carlo simulations is the compromise between coverage of the parameter's space and simulation time. The risk is to not detect possible critical configurations corresponding to very rare events, falling beyond the 3σ of the parameter PDF. To overcome this issue, an alternative way is to use advanced techniques which are able to fast isolate worst-case configurations that can be then tested in the high-fidelity full linear simulator, like the already cited structured singular values computation algorithms (also known as μ -analysis) in linear domain or based on integral quadratic constraints (IQCs) [6] in nonlinear domain.

Once the MIL validation phase is finalized, the next step is the SIL validation, where now the synthesized controller is translated in embedded-like code, in most of the cases in C/C++ or JAVA language. However, all the other elements of the MIL test are kept in the MIL environment. This

means, for example, that if the sensors/actuators, the environment, and the plant are implemented in a Simulink model, the controller/estimator is incorporated in an S-function where a conversion in C code of the control model is available. The code translation is now an automatized operation (autocoding): by following some rules in building the simulation model, the user can obtain the equivalent embedded-like code in a straightforward way.

If differences with respect to MIL test rise and do not meet the level of acceptable degradation, the reason has to be deeply investigated in order to go back to the modeling step and make the due modifications.

The following step of the V&V process is the PIL test, where the previous controller code is actually embedded in a dedicated processor whose performance is generally comparable to the ones of the final spacecraft on-board computer (OBC). This test allows the GNC engineers to verify that the synthesized controller can be run on a processor with limited performance compared to an on-ground computer. An example of critical test is the one involving complicated estimation and navigation algorithms where a huge number of system states are involved, vision-based routines or convex-optimization guidance algorithms where the processor has to run several optimizations without introducing destabilizing delays.

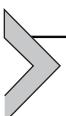
The final on-ground validation phase consists of the HIL test, where the real plant and/or the sensors/actuators are interfaced with the control loop running on the embedded processor.

For spacecraft applications, it is generally complicated to replicate on-ground the same conditions as in the Space. If some environmental parameters can be emulated as void condition, or Sun radiation, the microgravity condition cannot be by-passed in any way. The hardware elements that are generally tested in an HIL facility are then sensors and actuators like reaction wheels, while the spacecraft dynamics is simulated on a real-time platform. The commands to the actuators are then analogically sent by the real-time software simulating both the plant and the environment and computed by the control unit processor.

The final step of V&V activity is carried out when the spacecraft is on-orbit. All the hardware can now be verified in the operational conditions, and the expected system performance can be measured and compared to the mission requirements.

All the steps presented in this introduction, and extended in the following sections, do not proceed in a sequential way since several iterations are often needed before satisfying all the tests without coming back to previous steps.

In space history, the Hubble Space Telescope experience provides a clear example of how a V&V process can be invalidated in a very late phase of the project when unmodeled phenomena are not taken into account from the beginning of the GNC design. When the first version of the AOCS controller was implemented, an unexpected level of oscillation of the solar panel was observed on-orbit that was responsible of a degradation of the pointing requirement and consequently of the imaging quality. It was then discovered that this phenomenon was due to the thermal stress induced in the solar panels when the satellite was passing into eclipse. A new controller was then redesigned on-ground and sent to the spacecraft, and new solar arrays were installed during the first servicing mission [7].



Statistical methods

Before detailing the V&V steps of a GNC system, it is necessary to introduce the statistical methods necessary to perform such activity. In modeling the GNC system, designers shall take into account several uncertainties: mass and inertia values, eigenfrequencies and damping ratios for the flexible appendages, sensor noises, etc. All the physical parameters that can potentially have an impact on the performance of the system shall be taken into consideration with their associated statistical representation. The modeling of these effects involves the statistical interpretation of the phenomena, mainly to understand if the designer is dealing with a stochastic disturbance or, instead, a deterministic uncertainty. The word “stochastic” comes from the Greek *stokhos*, meaning “aim, guess”: therefore, stochastic refers to something randomic, in its own happening or in the amplitude of the phenomenon. In spacecraft GNC, this is the case of random errors on sensors, for example. On the other side, examples of limited uncertainties can be bias or low-frequency errors, or even the deviation of mass properties respect to the measured values.

Moreover, the variation over time of a low-frequency error can be associated to well-known periodic phenomena, such as the variation with temperature, which will periodically vary during an orbit in low Earth orbit (LEO) because of the alternation between sunlight and eclipse. Therefore, the most appropriate way to represent this kind of uncertainty is through a sinusoidal variation with period equal to the orbital period and amplitude specified by supplier data. If the amplitude of this variation is not well known, then a statistical consideration shall be carried out.

Other type of errors can be treated in a very similar way. For example, sensor misalignments due to actual mounting on the spacecraft, or even misalignments measurement residuals, can be represented by a number that will be constant in time but defined randomly at simulation start with its associated 3σ value (see [Chapter 6 – Sensors](#)).

The same goes for scale factors: they can be typically addressed as a Gaussian distribution in a well-defined range, for example, mean $\pm 3\sigma$ value. On the other hand, it can be more realistic to consider bias errors as picked randomly in a uniform distribution (see [Chapter 6 – Sensors](#)).

It can be well understood at this point that running a few simulations with error values set in a predefined way – randomly chosen in their possible range with their given distribution – is not fully representative. Indeed, in this way, just a limited number of random cases will be analyzed, and the obtained performance in this nominal subset is just a partial indication of the overall system performance. To have an extensive verification cycle at simulation level, stochastic analysis shall be performed to test the robustness of the GNC algorithms to the variation of external and internal conditions. To this extent, Monte Carlo analysis is performed, where a batch of simulations is run for the same scenario, scattering initial conditions in the appropriate range and through the appropriate distribution.

[Table 12.1](#) shows an example of a typical approach when dealing with a Monte Carlo analysis. For example, considering the first simulation set dealing with the release from launcher conditions, in order to simulate correct damping of angular rates and Sun-pointing acquisition, two different statistical representations will be considered:

- Initial attitude scattering: Sampled according to a uniform distribution since the satellite can have whichever attitude indifferently at launcher release.
- Satellite angular rate: Sampled according to a Gaussian distribution over a zero mean with the standard deviation value provided by the launcher manufacturer itself.

The previous considerations about sensors errors modeling shall be applied choosing the most appropriate statistical representation for each error contributor. The number of simulations for each scenario reported here is 1000 and is found targeting a failure probability (f) of 0.3% and a confidence level (C) of 95% using the following formula [\[8,9\]](#):

$$N. \text{ of simulations} = \frac{\ln(1 - C)}{\ln(1 - f)}$$

Table 12.1 Example Monte Carlo analysis schedule.

Simulation	AOCS mode	Number of simulations	Initial conditions			
			Attitude	Angular rate	Position in orbit	Sensors noise
1. Release launcher	from Safe	1000	0–360°, uniform distribution	0 ± X°/s 3σ on each axis	0–360°, uniform distribution	Gaussian, errors from datasheets
2. Image acquisition	Nominal	1000	Deviation from target: X°, uniform distribution	Nominal ± X°/s 3σ on each axis	0–360°, uniform distribution	Gaussian, errors from datasheets
...						

With the selected values for C and f , it is obtained a number of simulations of 997, then rounded up to 1000.

Other variables can be scattered either, such as the initial position in orbit or the mass and inertia values. Obviously, uncertainties representation reflects also in defining the uncertainty domain over which the GNC and especially the controller behavior is investigated, in terms of stability and margins: the more uncertainties, the wider the uncertain domain over which the controller stability shall be verified. For this reason, it can be convenient to not consider a very high number of variables to be scattered, and eventually complete the assessment with worst-case scenarios for a more comprehensive treatment of the extreme, but yet realistic, cases (worst-case analysis, WCA).

All the considerations done so far allow the GNC designer to better model the errors involved, and by doing so, to evaluate the system performance in a broader and more realistic way: the performance shall be then evaluated against requirements to assess whether the system meets the specifications.

Indeed, requirements involving the GNC subsystem will typically include statistical considerations in order to correctly assess the verification. For example, a requirement could state:

“The two-axis error between the actual and the target payload boresight direction shall be less than 100 arcsec 99% of the time.”

In general terms, the probability is expressed as a fraction or a percentage, while the expression $n\sigma$ is used only when a Gaussian distribution applies, as the concept of standard deviation does not provide additional information in a uniform distribution. The same requirement could be expressed in other forms, but the rationale behind it remains the same: considering the physical quantity X and a maximum value X_{max} , the probability of X being smaller than X_{max} is constrained to be higher than a certain value P :

$$\text{prob}(|X| < X_{max}) \geq P$$

This kind of high-level requirements is typically addressed through budgets: the contributing errors shall be identified and then characterized based on their statistical classification, usually summing up biases together and treating separately random errors with their standard deviations. Then, a root sum square can be considered in order to have a final result. A deeper discussion on control budgets is presented in [Chapter 10 - Control](#).



MIL test

Nowadays a baseline method to prototype space GNC software algorithms is to develop a simulator in MATLAB/Simulink environment following a model-based component approach. The models are validated in an MIL environment where the GNC algorithms are tested in closed loop with a simulation of the mission environment and spacecraft subsystems. The Simulink GNC models are then autocoded (by means of automatic techniques and tools), and the generated code is validated in real-time test benches where a prequalification testing campaign is performed to provide the GNC software ready for integration into an on-board software (OBSW) for complete software qualification. This development approach allows a fast GNC software design, V&V process that grants a reduction in the cost and time for producing a flight GNC software for a mission. The autocoding process is a fundamental step to be executed between MIL and SIL test. In this book, we will refer to autocoding during the description of the MIL test since, during this phase, it is crucial to develop models that are ready to be autocoded.

The code used in the space OBSW (e.g., mainly C or ADA) is very often categorized as critical by software standards (e.g., category B critical software as per ECSS [10]) due to the particular environment where it operates. Thus, specific aspects of the software development and verification shall be clearly defined and analyzed as well as the quality of the code produced according to software standards. This last aspect is particularly important if OBSW code is generated by automatic code generation tools and techniques starting from MIL environments.

Modeling of AOCS/GNC algorithms

The development of GNC algorithms follows a model-based design approach. This means that the complete GNC subsystem is implemented in a single design platform that allows the engineers or software developers to create visual simulation models of the complete system and check the proper working of the GNC algorithms before the implementation of the correspondent code (automatically generated or hand-written).

The most common testing platform used for GNC algorithms development is the MATLAB/Simulink environment. MATLAB is widely used since it allows fast prototyping and powerful analysis tools and functions.

Moreover, reuse of Simulink models already validated in other projects is extensively encouraged, as per model-based engineering philosophy.

The MATLAB/Simulink implementation could evolve in flight code using the autocoding tool chain, ensuring the compatibility between the design at GNC model level with the design at application software (ASW) level (i.e., code). This means that the GNC models allow capturing the core behavior of the final software product, description and verification already at specification level. In addition, it allows the developers to follow a smooth verification of the algorithms considering that the GNC models have been completely specified in terms of algorithms, interface, and parameters, and also tested and validated at unit, integration, and functional levels in the Simulink environment, allowing its reuse at ASW level. Please note that we refer to unit test when individual modules are tested individually in isolation, integration test if the different modules are tested working together, and functional tests when the whole system behavior is tested as per defined requirements.

The models play a fundamental role in the GNC development since the requirements and architectural design specification are captured and supported from early stages in MATLAB/Simulink simulators (e.g., 3 DoF simulators where only translational propagation of the vehicle is considered, and basic laws are implemented for the mission environment) that can be executed and provide feedback to the engineers for consolidation of GNC and mission requirements. Further developments can use the 3 DoF simulator as starting point to generate a full 6 DoF simulator with the complete GNC algorithms for attitude and translational control and more high-fidelity models of the mission environment and hardware units involved like sensors or actuators.

Architecture

The GNC algorithms are designed and implemented into MATLAB/Simulink environment in a functional engineering simulator (FES) or so-called MIL. This environment accounts for the GNC models (mainly guidance, navigation, control and vehicle management) and also for real-world models that provide the representative context for the validation of the GNC algorithms. The real-world includes models that are relevant for the GNC development and verification activities such as:

- Mission environment and perturbations (see [Chapter 3](#) – The space environment).
- Attitude dynamics propagation (see [Chapter 5](#) – Attitude dynamics).

- Translational dynamics propagation (see [Chapter 4](#) – Orbital dynamics).
- Sensors (see [Chapter 6](#) – Sensors).
- Actuators (see [Chapter 7](#) – Actuators).
- Equipment (affecting directly GNC like solar array drive mechanisms, deployment mechanisms, appendages, etc.).

Other equipment like thermal models, power models, telemetry/tele-command (TM/TC), transmitters, etc., are not generally modeled in the FES environment since they do not provide added values for the development of GNC algorithms and correspondent V&V activities.

In order to apply the model-based design and autocoding methodology, the MIL simulator shall clearly define separated subsystems for the GNC and the real-world models. Before the next V&V phase (SIL), the GNC will be autocoded (into typically C or ADA), integrated into the OBSW, and executed on-board during mission, while the real-world models are elements used only on-ground facilities for GNC testing and verification purposes.

Typical architecture for MIL simulator is represented in [Fig. 12.1](#).

The real-world models, simulating the environment and the sensor expected behavior, provide the sensor measurements to the OBSW. These measurements are preprocessed inside the OBSW and fed to the GNC. The GNC block receives the measurements, executes the respective GNC functions and provides actuators commands to guarantee the desired performance. These commands are sent back to the real-world where the actuators models generate the actuations considering hardware modeling aspects. The spacecraft dynamics is then executed and generates the state

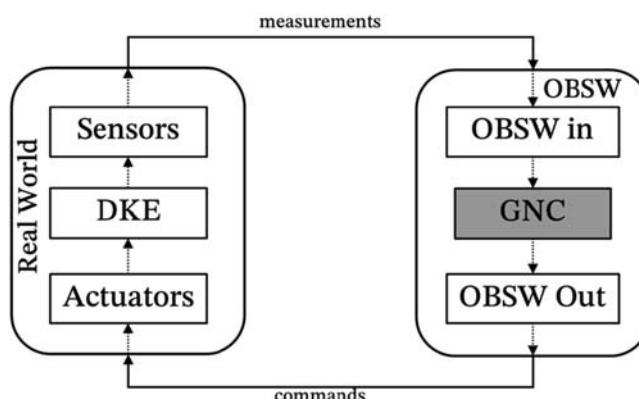


Figure 12.1 Typical MIL simulator architecture.

vector information (e.g., position, velocity, attitude quaternion, and angular rates) used by sensor models to carry out the measurements for the next execution cycle.

Please note that often a dynamics and kinematic environment (DKE) model is considered as including the environment, perturbation, and dynamics propagation models of the spacecraft. On the other hand, the GNC model is the ASW that will be executed on-board together with the complete OBSW. Even if only GNC block will be autocoded, a good practice is to implement in the FES environment also simplified models of the OBSW functions that are relevant for GNC correct behavior (e.g., OBSW in and OBSW out as in Fig. 12.1) that allow preparing the GNC input vector data for execution of a step of the algorithms (like creation of measurements' buffers, units conversion, routing of signals, etc.) and prepare the GNC output vector data to be sent to real-world models.

Avionics delays

The MIL environment is the first testing facility where the GNC is developed. The autocoding methodology allows generation of the code that will be ported incrementally to more space representative environments like PIL, HIL, or ATB (Avionics Test Bench). Thus, an important aspect of the MIL modeling is to consider already at early development stages the delays that will be faced when avionics equipment (i.e., hardware elements) is integrated in facilities like ATB. Delays are typically generated by the commanding and communication algorithms of the OBSW (e.g., OBSW scheduling function to command actuators during the execution cycle, encoding/decoding functions, etc.) and physical delays of the signal manipulation and transmission to the hardware elements. Thus, the control algorithms shall be developed to be robust enough to cope with these avionics delays and the GNC shall be validated considering the worst case of delays already at MIL level. An important aspect in modeling the avionics delays is that in Simulink implementation, the delay introduced shall be deterministic to guarantee a repetitive behavior of the GNC algorithms toward WCA.

Multirate

The GNC architectural design is generally organized by functionalities, typically a block for navigation, guidance, control algorithms and management logics. Even if this approach is compatible with autocoding, the most optimized approach is to organize the GNC according to execution frequency

of the functions and set the Simulink options for tasking and sample time to treat each discrete rate as separated task. The management of the multi-tasking and multirate factors will determinate the software integration approach of the generated code with the OBSW and the real-time operating system (RTOS). Note that, in the OBSW, all sample times shall be an integer multiple of the model's fixed step size. Following the rates organization approach, clear separated functions will be directly generated from autocoding associated with each execution rate of the GNC models. Then, the management of integration and scheduling of the production code with the rest of OBSW is responsibility of the software integrator. In fact, these factors will affect the scheduling implementation of the basic software, which is the part of the code that manages the low-level functionalities or basic services like scheduling, data interchange between modules, etc. This way, the GNC software, which has been divided into different functional units (previously identified), will be controlled by the OBSW and the on-board RTOS.

Tunable parameters

In Simulink environment, the GNC models are configured via parameters that will determine the tuning of the GNC itself. These parameters can be modified and updated in the MATLAB workspace to execute different tests during simulations campaign. The same mechanism is desirable for the auto-generated GNC software where the OBSW should provide functions or a method to update the mission parameters. When generating the code, if not specified otherwise, the GNC parameters will be hard-coded. This aspect generates the need to define the parameters as tunable in the Simulink environment to have the possibility for updates also in the generated code. A typical approach in GNC development is to group parameters as tunable and nontunable sets. This solution allows to quickly identify which parameters shall be mapped to variables in the code with the possibility of updates from OBSW functions that provide the capability to modify specific memory areas of the data pool (group of data that can be accessed from OBSW functions). These parameters shall be set as tunable in the MATLAB/Simulink environment (see MATLAB help for tunability [11]). Strictly speaking, autocoding is not part of the MIL test (where only MATLAB/Simulink simulation are run). However, MIL models shall be developed following autocoding best practices to avoid useless iterations on the models' design. For this reason, the most important aspects in MIL modeling

toward automatic generation of the GNC software are resumed in the following points:

- Clear and separated interfaces must be defined between the real-world and the OBSW models.
- Interfaces of GNC block shall define size, data type, and sample time. All these properties of the import and output blocks shall be explicitly defined. No Simulink inherited (-1) settings shall be used.
- The Simulink simulator must use discrete solver and time steps. The GNC software will be embedded into an OBSW executed in an OBC working at discrete times.
- Explicit definition of tunable parameters.
- Not use of mask initialization function.
- Architectural organization of GNC as per frequency.

Requirements tracing

The GNC requirements drive the models development and implementation. A set of user requirements is specified at the beginning of the software lifecycle, and it is used to describe the complete behavior of the algorithms implemented by the models. Frequently, the models developed in MATLAB/Simulink are used as a requirement specification itself due to the ease of generating simulations (open loop and closed loop) that demonstrate the correct implementation of the algorithms.

In this context, the MATLAB/Simulink environment is used for model development and for the verification of the models' requirements. Simulink can be extended with add-on products for verification, validation, and testing activities that enable a continuous testing and verification throughout the development process.

The requirements typically stored and managed by a tool (e.g., DOORS) can be linked to the correspondent model implementation in Simulink. The link shall track univocally the requirements ID with the related models (e.g., textual link, hyperlink, etc.). A modeling verification tool may be used to define and manage the links created in the Simulink models and to verify their validity.

GNC optimization

At this step of the FES development, some preliminary work of identification and analysis of the Simulink subsystems containing the main GNC modules and their interfaces with the rest of systems (e.g., OBCSW and real-world) shall be performed, as shown in the example reported in Fig. 12.2.

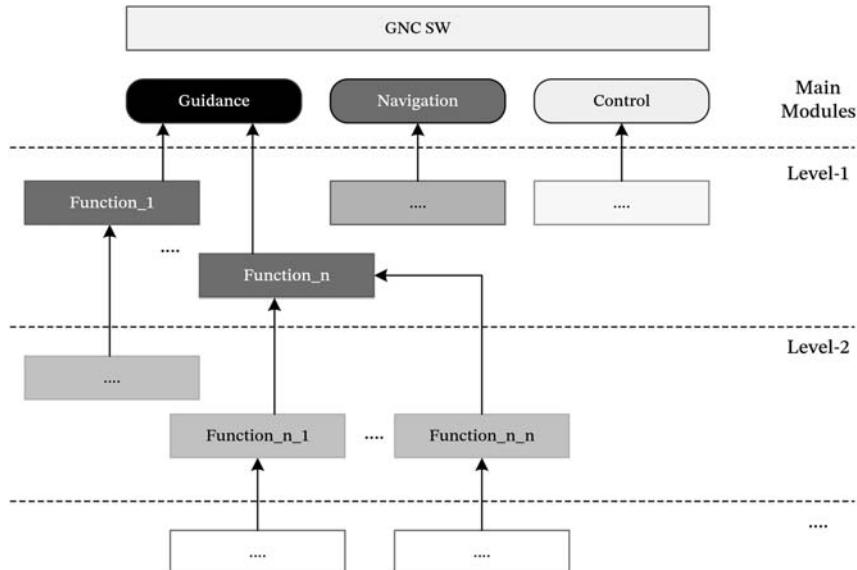


Figure 12.2 Example of GNC decomposition and functions identification.

The optimization of the architecture of GNC consists of the following main activities:

- Identification of the functions that define the GNC and setting them as reference models or atomic subsystems. The GNC ASW design is started by creating a root class representing the overall GNC system, and then it is decomposed into smaller pieces according to the simulator architecture and following the autocoding methodology inherited from the use of the autocoding tool (e.g., Embedded Coder [12]).
- The isolation of the generated code functions is based on:
 - Isolation of main functionalities.
 - Isolation of computational critical elements (to easily test computational bottlenecks).
- Limitation of the generated code size per function and module (code file). A maximum number of basic blocks shall be defined in order to limit the code size for each function (control metric).
- Identification of the working rate and frequency of each component.
- Identification and definition of the interfaces of each component.
- Code specification using autocoding tool (e.g., Embedded Coder [12]) properties inside of the selected Simulink subsystems (where previously have been defined/declared the variables, data structures, scaling formulas, function calls, tasks, etc.).

Modeling rules

The use of autocoding techniques to generate code from a model-based development (e.g., via MATLAB/Simulink) implies the use of rules and guidelines to guarantee the quality and compliance of the produced code with the required standards. Again, this is not necessarily part of the MIL test, but it is reported here as a fundamental step during MIL models generation.

The guidelines for the GNC modeling in MATLAB/Simulink may be grouped into two categories:

1. Modeling architectural and design rules

This group defines the rules/guidelines that need to be followed at architectural and design level of the GNC subsystem. These rules allow the automation of the real-time validation process in an integrated environment and ease the porting of the generated code to the SIL and PIL test setups without significant modification of the Simulink models and guaranteeing architectural mapping between the Simulink models and the produced code.

2. Modeling implementation rules (coding and style)

This group defines rules/guidelines that need to be followed by the Simulink implementation of the GNC models. These rules prevent errors, language-specific pitfalls, nonoptimized statements, forbidden constructs, complexity restrictions, and readability. These rules primarily increase the reliability of the validation process and allow the generation of optimized code.

These modeling rules shall be taken into account from the starting of the GNC development (as they imply architectural constraints, development strategies, etc.). Moreover, these rules must guarantee that the proper level of numerical accuracy is obtained in the generated code. In order to do it, the requirements and guidelines shall be checked and implemented through the use of a checking rule tool (e.g., Simulink Check toolbox [13]), which permits to define coding style and rules and performs automatic checks of the Simulink models (e.g., through the Model Advisor [14]). For more details, please refer to [Chapter 18](#) - Autocoding best practices.

Verification activities at MIL level

The main activities related to the GNC modeling V&V are listed below:

- Algorithms V&V.
- Requirements verification.

- Modeling standards verification.
- Model coverage verification.
- Profiling of the models.

Algorithms verification and validation

The functional validation of the GNC algorithms is performed in MIL environment via specific tests, sensitivity analysis, or Monte Carlo analysis. This validation is performed against the GNC requirements. Please see specific section on Statistical Methods.

Requirements verification

The requirement verification activities for the GNC software are executed at two levels:

- Models requirement verification.
- Code requirements verification.

Models requirement verification

The requirements that have been used to the drive the development of the software shall be verified in the models implementation in Simulink environment. The following steps are considered:

1. Link the requirements to Simulink models.
2. Verify the requirement trace from Simulink models.

The requirements implemented in the Simulink models shall be verified. A traceability matrix or a report shall be created to assess the requirements that have been considered in the Simulink implementation. A modeling tool may be used to generate automatically the traceability matrix and verify the requirements coverage at model level.

Code requirement verification

The requirements that have driven the software design at model level in Simulink environment shall be maintained in the production code generated. The same steps used for models requirement verification are considered but at code level:

1. Link the requirements to production code. The requirements, in this case, shall be linked to the correspondent implementation in the generated code. The link previously defined at model level during design and implementation shall be maintained at code level. The code generator tool shall provide the capability to keep the track of the requirement during automatic code generation.

2. Verify the requirement trace from production code. The requirements implemented in the production code generated shall be verified. As before, a traceability matrix or a report shall be created to assess the requirements that have been considered in the production code. A coding tool or directly the code generation tool may be used to generate automatically the traceability matrix and verify the requirements coverage at code level.

Models profiling

A preliminary model profiling may be also performed on MATLAB/Simulink to obtain raw numbers (mainly relative numbers between trade-off solutions), and then, these parameters will be used in the GNC algorithm trade-offs and implementation solutions. Profiling the algorithms will help in finding potential bottlenecks from a computational point of view, and it is a preliminary analysis for performing an optimization in the implementation of the models if any issue is detected. This analysis will provide a preliminary estimation of the CPU consumption of the GNC algorithms.

Modeling standards verification

The development of GNC software shall be compliant with requirements, and at the same time, it must consider also design and implementation aspects for guaranteeing a smooth and easy code generation using autocoding techniques.

A set of rules and/or restrictions at modeling level can be defined or tailored from standards for specific mission to ensure later generation of high-quality code. V&V activities shall be performed on the Simulink models to assure that the modeling guidelines have been respected at design and implementation level.

The verification of modeling rules and standard on the Simulink models can be performed since the beginning of the software design phases supporting the models' development for the autocoding. The following activities are defined for modeling rules verification:

1. Select the rules or standards to be verified. The rules and standard to be verified at model level are specified (or agreed with the customer) for the particular project. Predefined standards may be used or even project specific rules may be defined. Among the most common coding standards, we can find:

- Motor Industry Software Reliability Association (MISRA) [15]: a set of C/C++ coding guidelines ensuring code safety, security, and reliability in embedded system software.
 - MathWorks Automotive Advisory Board (MAAB) [16]: a set of modeling guidelines for the usage of MATLAB, Simulink, Stateflow, and Embedded Coder.
2. Run the checks on the models. The rules and standard are verified over the Simulink models running the scripts that implement the check of the rules. A modeling verification tool is typically used to run automatically the checks specified.
 3. Analyze the report of pass/fail checks. A report detailing the rules passed and failed during verification is produced and analyzed in order to assess the models compliance with the rules and standard selected. A modeling verification tool typically generates reports from the checks run.
 4. Update the design/implementation to correct the checks failed. The failed checks drive the software designer to update and modify of the Simulink models affected. These updates may be performed manually (e.g., typical design updates that require human analysis) or automatic scripts may be also provided by the modeling verification tool (e.g., typical implementation updates that are well identified).
 5. Rerun the checks. After the models updates, the checks of the rules are run again to verify the correct modifications. The process can be repeated till the rules and standard checks are totally passed or justified.

Model coverage verification

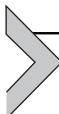
Unit tests, integration tests, and validation tests are defined to validate the GNC algorithms implementation from functional and interfaces point of view. The model coverage is also an important aspect in model validation since it provides the evidence of how many modules and subsystems have been really executed during testing campaign. On this aspect, the percentage of model coverage has to be agreed with the quality standards to be followed in the GNC development.

Anyway, when possible, 100% of model coverage is desirable since it is a considerably useful intermediate step to reach the 100% of the generated code coverage. In fact, definition of the coverage tests for the models is setup directly in MATLAB/Simulink environment and can be totally reused to verify code coverage in SIL simulations.

Model coverage is considered to be reached by cumulative coverage percentage all over the unit, integration, and validation tests executed in MIL configuration.

The verification of model coverage can be performed according to the following steps:

1. Select the coverage type to be verified. The type of coverage analysis to be performed over the Simulink models (e.g., decisions, conditions, MC/DC, etc.) is specified and agreed with the customer. A modeling verification tool is typically used for model coverage that is configured to execute the specific coverage analysis selected.
2. Run the coverage on the models. The tests defined for the Simulink models (e.g., unit, integration, or validation tests) are executed. The modeling verification tool instruments the models to get the coverage results for the tests run.
3. Analyze the report of coverage. A report detailing the percentage of the models covered by the tests (for the type of coverage specified) is produced and analyzed. The modeling verification tool typically generates reports that can show directly in the Simulink implementation the blocks/paths covered.
4. Add tests to increase coverage if not target percentage. If the target percentage (to be agreed with the customer) of model coverage is not reached, new tests shall be defined. The analysis of the coverage report drives a definition of new set of data and tests to force the execution of all the uncovered blocks contained in the models. Thus, additional tests are created in Simulink environment (MIL).
5. Rerun the model coverage for the additional tests. The additional tests are run, and the model coverage is verified again (e.g., cumulative coverage analysis). If 100% of coverage is not reached, yet the process can be repeated till the model coverage is 100% (passed by tests or justified) or target percentage is reached.



SIL/PIL test

As already mentioned in the previous section, typical GNC software design, development, and verification strategy is based on autocoding of the GNC MATLAB/Simulink models in order to generate code optimized for embedded systems. After GNC algorithms verification at model level, the autocoding is performed and testing of the GNC generated code is executed in the SIL environment aiming to verify that the GNC code

behaves like the models. The following step in the verification process is to execute the code in a PIL test bench that computes the GNC code into OBC to verify performance and real-time aspects of the software. The following sections describe the V&V activities performed in SIL and PIL environments.

Autocoding

The GNC development strategy is part of an integrated, coherent, and incremental approach based on the chain MIL → Autocoding → SIL → PIL → HIL that allows porting the GNC algorithms from Simulink simulators to hardware-embedded systems (Fig. 12.3) in a consistent and incremental process.

Autocoding of MIL-validated GNC models is performed to produce code and start the GNC software V&V process. An SIL environment is the first step where the generated code is functionally tested by integrating it into the MIL simulator in MATLAB/Simulink environment before going to the PIL or HIL test benches. The objective of the SIL environment is to verify that the generated code functionally behaves in the same way (apart numerical accuracy) with respect to the algorithms implemented by the models. Further step in GNC software validation is to execute the code in a PIL test bench in an OBC to verify performance and real-time aspects of the software.

This V&V approach chain can provide very productive support during the development phases and possibility to test requirements already at early and intermediate design phases. This approach allows fast iterations and feedback and the possibility to correct design problems from the beginning of software development minimizing the required effort.

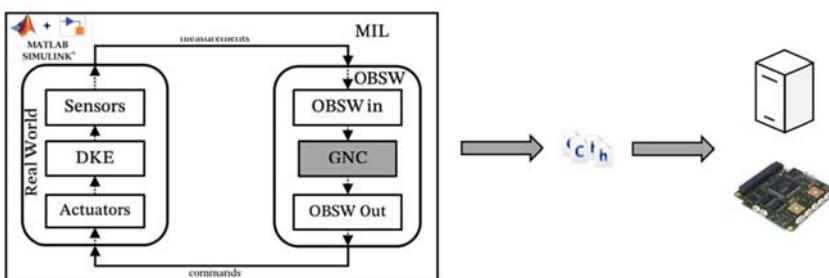


Figure 12.3 Porting GNC models to embedded systems.

The autocoding methodology shall reflect all the design, development, verification, and validation aspects of the software lifecycle, in particular, for GNC software, where the autocoding techniques and tools are involved. Fig. 12.4 shows the main activities related with autocoding that have to be performed on the GNC software from the design to validation phases starting from requirements down to model and finally to code in order to produce flight software.

Autocoding activities are considered at three levels: requirements, Simulink models, and production code. Most of the steps have been already described for MIL test but are summarized here for the sake of clarity:

1. Requirements:

- o Support Simulink models implementation. The GNC requirements drive the models' development and implementation.
- o Trace the requirements to Simulink models. The GNC requirements are generally stored in a database, and they must be traced to the correspondent Simulink models to justify the implementation.
- o Trace the requirements to code: The GNC requirements must be traced also to the code generated automatically from the models.

2. Simulink models:

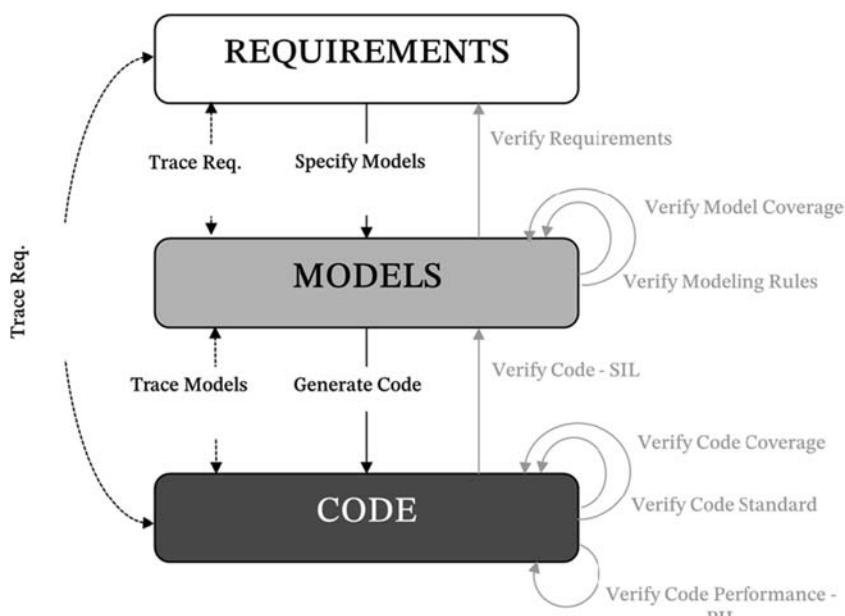


Figure 12.4 GNC software autocoding methodology main activities.

- o Trace Simulink models back to requirements. The Simulink models must be traced back to the correspondent set of requirements that justify the implementation.
- o Verify requirements. Verification of the requirements trace down to Simulink implementation must be produced.
- o Verify modeling rules. The GNC model-based development shall follow several rules and guidelines for allowing the compatibility of the Simulink models with the autocoding process for guaranteeing the quality and compliance of the produced code with the required standards. Verification of the modeling rules in Simulink implementation shall be produced (statics analysis on models). Reports can be generated for the Simulink models to assess compliance with rules and standards.
- o Verify model coverage. The GNC models implementation shall be verified by tests (dynamic analysis on models). These tests are focused to validate the algorithms, but tests shall be also specified to reach a target percentage of model coverage.
- o Generate code. The Simulink GNC models are converted automatically to production code (e.g., C or ADA). A code generator tool is properly configured to generate an optimized production code (e.g., metrics, memory, complexity, readability, etc.). A code generation report may be also generated to support the models architectural mapping and traceability to code.

3. Production code:

- o Trace code to models and requirements. The code generated must be traced back to the correspondent Simulink models and back to the set of requirements that justify the implementation.
- o Verify code—SIL. The behavior of GNC code generated shall be maintained with respect to the Simulink models. The same software tests performed first at model level are then executed at code level through SIL configuration (embedding the code generated automatically into the FES simulator as S-Function) to verify the correct generation of the code (same tests result for models and code).
- o Verify code standard. The GNC code shall guarantee the metrics and efficiency of the code generated via autocoding. The verification of a coding standard shall be produced for this purpose.
- o Verify code coverage. The GNC code shall be tested and typically 100% of code coverage shall be reached as requested by the standards for flight software (e.g., ECSS [10]).

- Verify code performance—PIL. The real-time and profiling characteristics (e.g., schedulability, memory budget, worst execution time, etc.) of the GNC-generated code are verified by PIL tests. A specific target shall be selected to run the generated code in the PIL configuration.

Please note that the steps above are defined to generate prequalified GNC flight software. For testing or demonstrating purposes where the GNC software is only executed in ground facilities, some steps may be relaxed and shall be agreed with the customer based on project needs and budget.

Moreover, settings of code generator and Simulink subsystem options shall be agreed with the GNC software integrator. The way the OGSW executes the GNC affects the interfaces and the functions to be generated depending on frequencies (i.e., scheduling of the functions based on rates where a single-entry point function is generated for each rate) and management of GNC input/output vector and tunable parameters (i.e., way to update the data pool that refers to GNC data).

Software-in-the-loop

When autocoding is used, the GNC V&V is based on a stepwise approach starting from MIL to SIL, porting the Simulink models to code. The GNC software is produced from the models according to the code generator (e.g., Embedded Coder [12]) settings selected for the project that in general must be agreed with OGSW integrator. Typically, an SIL block is generated as well from autocoding process. This block is a Simulink S-function configured to execute the generated code so that the GNC can be computed in close loop, as shown in Fig. 12.5. This S-function can be automatically integrated into the MIL simulator (replacing the GNC model or executed in parallel) to verify that the generated code provides the same results as the original Simulink model. The new simulator containing the S-function is executed using the complete MIL simulator inputs (including real-world), and the results are compared with respect to the reference results obtained during the validation campaign of the MIL.

Processor-in-the-loop

After SIL testing, an additional step in the GNC software validation is to execute the code in a PIL environment. The GNC-generated code is integrated into a software which implements functions related with GNC

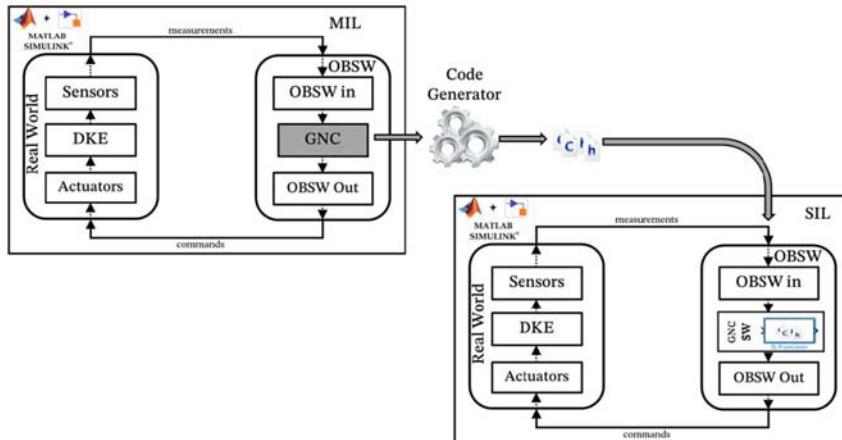


Figure 12.5 From MIL to SIL.

execution (i.e., preparing the GNC input vector data from measurements, execute the GNC step function, and prepare the GNC output data for commanding). This ASW is compiled using the same flags and options that will be then used for complete OBSW. These settings must be iterated and agreed with the OBSW integrator. The compiled binary image is loaded in a RTOS (e.g., RTEMS [17]) that runs on the real OBC (e.g., LEON board, ARM based processor, etc.—See Chapter 13 - On-board implementation) or even on a target processor simulator (e.g., TSIM, TS, etc.) that allow the emulation of different real processors behavior. The PIL test bench allows testing the software in flight realistic conditions regarding the space representative processor and with simulated environmental conditions that could also be autocoded from FES/MIL (e.g., DKE, actuators, sensors, etc.) and ported to a real-time environment, as in Fig. 12.6. The PIL is the first test bench used to perform the integration between the GNC software (autocoded) and the data handling and execution management software (hand-made code).

Verification activities at SIL level

The main activities related to the AOCS/GNC modeling V&V are listed below:

- Code functional verification.
- Requirements verification.
- Code standards verification.
- Code coverage verification.

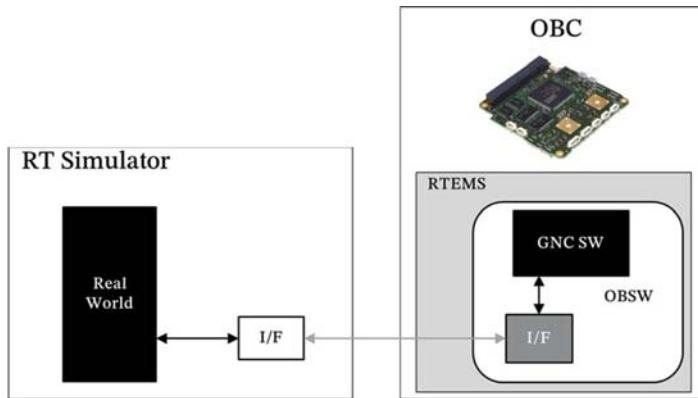


Figure 12.6 PIL environment configuration.

Code functional verification

The GNC production code follows a development totally dependent of the model design in term of specification, design, coding, unitary test, integration test, functional and performance validation, since it is based on autocoding tools (e.g., Embedded Coder [12]).

Considering autocoding and model-based design for GNC software, all these tests are performed first at model level and then they have to be executed also at code level as well through SIL (embedding the code generated automatically into the Simulink simulator). Currently, no code generation tool can guarantee that code generated from MATLAB/Simulink behaves (at least from functional point of view) exactly the same of the model, avoiding in this way the execution of unit and integration tests on the generated code (as required by software standards [10]).

The GNC software tests are defined and executed to validate the models in an MIL configuration in MATLAB/Simulink environment. The same tests, models, and additional test environment can be autocoded and then executed in SIL configuration. The SIL allows the verification of the code functionalities, interface, and coverage. A set of reference tests are selected from MIL simulations (e.g., typically specific cases from Monte Carlo analysis) and executed in SIL environment. Reference outputs from MIL execution are compared against the SIL results, as in Fig. 12.7. If the outputs difference is less than a defined threshold, the generation of the code is considered successful.

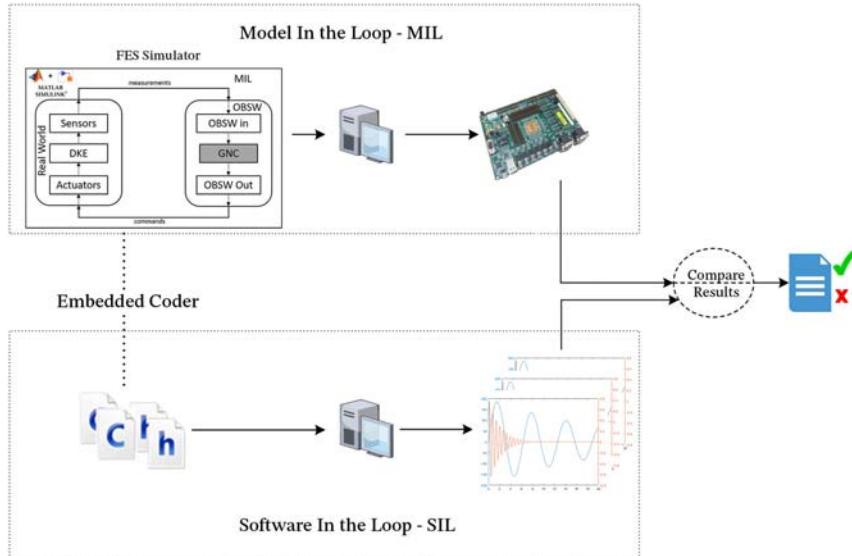


Figure 12.7 GNC software verification in SIL (MIL vs. SIL).

A threshold shall be considered in the comparison of results of MIL versus SIL since numerical accuracy is different between models and code. A final set of validation tests is also ported from SIL to PIL.

Requirements verification

The requirements that have driven the software design at model level in Simulink environment shall be maintained in the production code generated. Thus, requirements verification activities are also performed at code level in the same way they were verified at model level.

Code standards verification

Coding standards and metrics can be verified on the GNC code generated automatically from Matlab/Simulink models. Applicable standards shall be agreed with the client in the context of a project, but MISRA-C 2012 [15] is often proposed to be the standard that guarantees the metrics and efficiency of the code generated via autocoding. In fact, this version of the MISRA-C standard includes a number of improvements that can reduce the cost and complexity of compliance while aiding consistent and safe use of C in critical system. Moreover MISRA-C 2012 includes MISRA AC AGC—Guidelines for the application of MISRA-C 2004 in the context of automatic code generation.

The verification of coding rules and standard on the auto-generated production code is executed according to the following steps:

1. Select the rules or standards to be verified. The rules and standard to be verified at code level are specified (or agreed with the customer) for the particular project.
2. Run the checks on the generated code. The code standard is verified over the generated code via a static analysis. A coding verification tool is typically used to run automatically the checks to verify the rules of the standard.
3. Analyze the report of pass/fail checks. A report detailing the rules passed and failed during verification is produced and analyzed in order to assess the code compliance with the rules and standard selected. A coding verification tool typically generates reports directly in the code pointing out (e.g., with colors, markers, underlines, etc.) the lines that do not comply with the standard.
4. Update the Simulink model design/implementation to correct the checks failed. The failed checks drive the software designer to identify and update the implementation of the Simulink models that generated those lines or to update the configuration and settings of the code generator tool to produce a better optimized code compliant with the standard.
5. Regenerate the code from Simulink models. The code is generated again after the update of the Simulink models and/or update of code generator tool settings. The autocoding process is applied, and the code is produced.
6. Rerun the checks. The rules are run again over the new code to verify the correct updates, and the process can be repeated till the rules and standard checks are totally passed or justified.

Code coverage verification

The code coverage analysis is performed based on MIL to SIL conversion approach that guarantees that the algorithms of the GNC are correctly converted to production code by the code generator tool (i.e., validation tests passed at SIL level) and coverage that is tested at model level can be assessed also at code level. The tests defined in Simulink environment to assess model coverage are reused in SIL configuration to verify the percentage of code coverage. This approach allows reusing tests defined at MIL configuration and verify the efficiency of the code generator tools and/or settings. The percentage of code to be covered depends on the category of the code

generated and shall be agreed with the customer. For example, according to ECSS standards [10], if code is declared as category B, the 100% of the code coverage must be reached and this means that all the code statements and branches must be covered. But this level of coverage may be relaxed for code used just at test environment levels on ground (i.e., prototype code).

The verification of code coverage can be performed according to the same steps detailed for model coverage verification.

Verification activities at PIL level

The PIL test benches allow to test the software in space representative processor and with simulated environmental conditions. The PIL testing is the first real-time test bench where the final GNC flight software will be used in real-time closed-loop operation.

The main objective of the PIL campaign is to functionally verify the implementation of GNC software generated via autocoding techniques in a real-time closed-loop test against the requirements, mainly for performance, safety, robustness, and real-time aspects. Thus, the following activities are considered for PIL tests campaign:

- Assess the GNC algorithms robustness to variations in floating point implementations using real processor (simulated or hardware board), real-time OS, and cross-compiler.
- Assess code performance: computational load and memory and code size.
- Identify and check time margins in the data communication and algorithm execution on the real processor.
- Assess the algorithm's robustness to delays coming from synchronization between real-world and GNC application.
- Confirm that the integration of the GNC code inside the real-time OS of the OBC has been correctly performed.
- Testing behavior that cannot be tested in a modeling environment (for instance, to check optimized code, ANSI compatibilities, overflow protections, etc.).

Comparison of the performance results with results from the SIL testing campaign is a further validation milestone since it provides verification that the flight software implementation is compliant to the user requirements in a real-time environment using real or representative processor.



HIL test

The final on-ground validation step is represented by the HIL test. Contrary to PIL test, where only the GNC software is running on representative hardware, during HIL test, real equipment, such as sensors/actuators, is interfaced with the software running on the embedded processor.

Before the HIL test, different, sequential tests are implemented allowing to verify the hardware correct behavior:

- Interface and functional tests (IFTs). The hardware equipment to be tested is connected to the OBC and to the power supply to switch the equipment on and command its basic functionalities in an open-loop setup.
- Functional after integration (FAI). The hardware equipment is tested once it is integrated on the spacecraft flight model, therefore involving flight OBC, flight power control unit and flight connections that are supposed to be not disconnected in the following of the AIV campaign, otherwise the test is invalid.
- Full functional tests (FFT). It is aimed at being a quick verification of the correct basic functionalities of all AOCS/GNC equipment once the spacecraft has completed its integration and can be repeated several times after each major event that could have a potential impact on the satellite (mechanical loads test, thermal vacuum chamber test, transportation to the launch site, etc.).
- HIL. It is a verification technique involving a GNC software or equipment to be tested within a representative environment, in order to evaluate its performance in a closed-loop scenario with real hardware equipment.

Therefore, IFT, FAI, and FFT differ from the HIL tests for being tests performed on the hardware equipment without having a complete closed-loop scenario. Indeed, in hardware tests rather than HIL, no electrical ground support equipment (EGSE) simulating the environment and the plant is involved; therefore, the scope and target of these tests is limited to hardware functional behavior. As a consequence, HIL tests are necessary for a wider and more appropriate verification of performance requirements, as described in the following paragraphs.

In AOCS/GNC applications, HIL tests are used for two main purposes: verification of sensors or actuators in a closed loop, and, more often, verification of GNC software in a closed loop with representative hardware. The first kind of HIL test is performed with the real sensor or actuator

engineering model or even flight model and testing its functionalities. Performances and interfaces with the platform and the software running on the OBC through a closed loop typically composed of the unit itself, the OBC with the software, power suppliers, and an EGSE to simulate plant dynamics, environment, and missing equipment are verified. This kind of test is performed in order to anticipate the verification of the hardware and its interfaces with the system and the software, to avoid costly iterations, since bug-fixing is cheaper, the earlier the bugs are spotted, and thus their resolution can be applied in the process. On the other hand, the second kind of HIL refers to a similar setup but used for a different purpose: in this case, in fact, the unit under test is the GNC software itself, in a closed loop with the OBC (where the software runs), power suppliers, real hardware sensors, actuators, plant dynamics, and TM/TC interfaces, and of course appropriate wiring. This kind of test is performed to verify appropriate behavior and performance of the GNC algorithms in complete end-to-end scenarios, typically analyzing all the AOCS modes and simulating every possible transition between them. Therefore, for HIL related to hardware verification, the added value respect to the previous verification (MIL, SIL, and PIL) can be found in verifying the correctness of interfaces between the hardware under test and the system being simulated, including the software interfaces for complex equipment such as star trackers or GNSS receivers, i.e., the ability of the platform software to correctly command the equipment and retrieve its telemetry, and sensors or actuators performances. For the HIL aimed at verifying the GNC software, the added value relies in the verification of complete scenarios in a flight-like configuration that includes different features not present at model-level (run of the flight code on real OBC, integration with the system software, compiling, ...) nor at SIL/PIL level (interface with real, representative hardware, integration with the system software, dynamic scenarios instead of predefined input/outputs).

From the description provided so far, it should result clear that HIL testing is performed in real time, so that operations are executed or simulated in scenarios as close as possible to reality. Therefore, such a test becomes implicitly a verification of the correct execution timing of the tasks of all the equipment involved with its relevant software.

Examples of HIL testing for hardware verification

As mentioned, HIL test is used for equipment performance and interface verification to anticipate every possible aspect of the verification cycle that

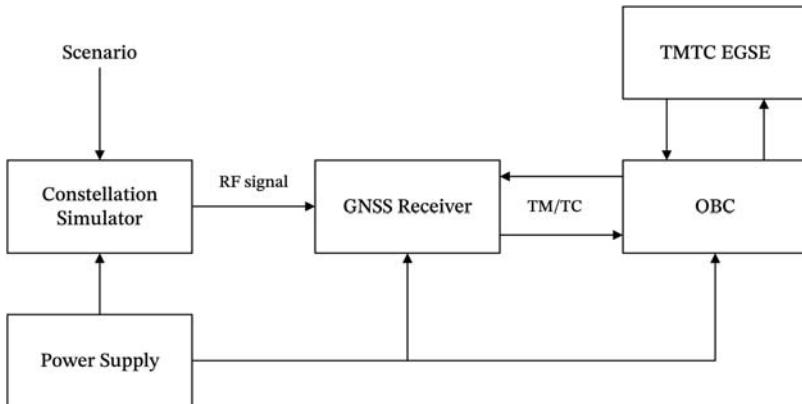


Figure 12.8 Example GNSS receivers HIL test set-up.

can cause impactful issues in the project. As an example of a GNC verification framework, one common HIL test is the performance test of GNSS receivers with a GNSS constellation simulators. The test setup, thought here for an Engineering Model, is depicted in Fig. 12.8: the Telemetry and Tele-command (TM/TC) line of the receiver is connected to the OBC, where the spacecraft software runs, and the power line is connected to a common laboratory power supplier. The constellation simulator is connected through RF to the receiver; therefore, the antenna is by-passed, but this typically doesn't prevent the test to be representative, since antennas are passive components; the OBC is connected to a TMT EGSE that permits to send telecommands to the system and retrieve telemetries, constituting therefore the Man–Machine Interface (MMI).

In this case, a predefined scenario is loaded into the constellation simulator, indicating the position in orbit of the spacecraft and its attitude during time, so that realistic visibility of the antennas toward GNSS spacecrafts can be simulated. At the end of the simulation, it will be easy to compare the true orbital position of the spacecraft (i.e., the one loaded in the scenario) versus the estimated one by the GNSS receiver, available as telemetry through the OBC. Therefore, this kind of test is extremely useful to evaluate actual performance of the GNSS receiver in realistic scenario and is frequently used by GNSS receiver producers to verify the performance of their product during the qualification campaign.

Another typical HIL test is the verification of the star tracker in the loop: in this case, a simulator is needed, in order to represent the stars in the sky as can be seen by the star tracker field of view. With this setup, it is possible to

conduct open-loop tests, or, if the star simulator is dynamic, even closed-loop tests but with predefined scenarios (predefined attitudes vs. time). For more complex and more exhaustive tests, an AOCS Special Check-Out Equipment (SCOE) capable of simulating the dynamics of the satellite should be included in the loop, so that complete end-to-end scenarios could be simulated, involving a dynamically evolving scenario according to the control action, where the star tracker EGSE takes the “real” quaternion simulated by the AOCS SCOE dynamics model and convert it into a video of the stars in the sky to be perceived by the star tracker itself, that will provide, in this way, an attitude solution as though it was in orbit in flight conditions.

Examples of HIL testing for software verification

A typical setup to verify the GNC software is shown in the schematics in Fig. 12.9: the GNC software runs on the OBC that is connected to real sensors and actuators (or only some of them) providing control commands to actuators and receiving sensors measurements and telemetries, and to the TMTC EGSE through which the test conductor can send telecommands to the software and retrieve telemetries. A power supplier and appropriate wiring complete the setup.

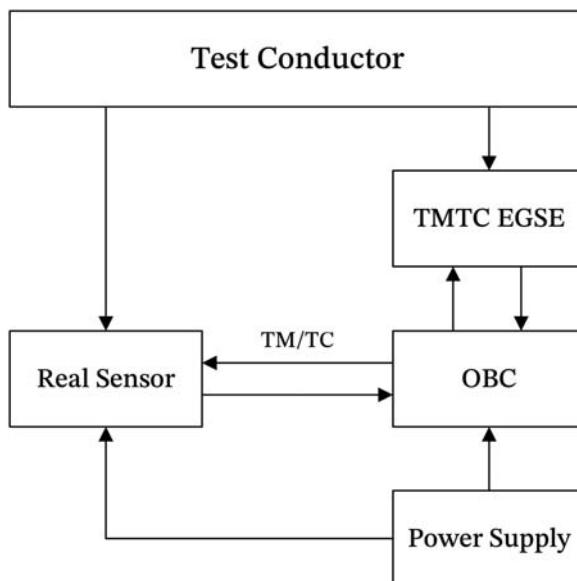


Figure 12.9 Example GNC software HIL test set-up.

For this architecture, spacecraft dynamics, the environment, and sensors and actuators interfaces are carefully modeled to allow representative closed-loop simulations. In details, the main blocks are:

- The software simulation part, composed of the DKE models, and sensors and actuators models of the ones that are not interfaced at hardware level with the GNC software. The dynamics and kinematics represent the plant behavior of the spacecraft in terms of attitude motion and linear motion, while the environment models represent Earth gravity model, Sun and Moon position propagation, the action of disturbances such as atmospheric drag, solar pressure, magnetic interferences, occultation of sensors field of view, etc. On the other hand, sensors models are aimed to reproduce the realistic behavior of boarded sensors for which the hardware is not connected to the OBC (see [Chapter 7 - Actuators](#)). Similarly, actuators models represent how the real actuators treat the command signals to provide a final control action acting on the plant dynamics, thus including, for example, in the case of a reaction wheel, jitter, ripple, friction, stiction, etc. (see [Chapter 7 - Actuators](#)).
- The hardware interface part manages the signals frequency, since the models in the software simulation part can run at whichever frequency is desired, but then a real-time conversion is needed. This is done for all the sensor/actuators for which the hardware is not connected to the OBC. Then, the software signals of sensors and actuators models are transformed in their equivalent physical signal sent to the proper electrical connector in order to allow a flight-like connection between the OBC and the equipment. In the case of sensors, signals will be sent from the hardware interface to the OBC, while in the case of actuators, command signals from the OBC will be retrieved by the hardware interface to be used for impacting the plant dynamics. In this phase, also the proper telecommunication protocol is simulated in order to feed the flight software of the spacecraft with signals as realistic as possible.
- Real hardware sensor and actuators are used to stimulate the GNC software with representative signals and commands. In this case, the hardware interface is not needed since the real hardware is directly used and interfaced with the OBC.
- The MMI, that is the graphic interface, visible on the screen of a computer, through which the test conductor can command initial conditions for the simulation, equipment switch on and off, adjust parameters, and see in real time the behavior of the simulated spacecraft in terms of attitude, angular rate, orbital position and velocity, etc., and the answer of the sensors. Moreover, a log of data is typically allowed for postprocessing.

This test setup is extremely useful for the verification of GNC software on-ground, since complete end-to-end scenarios, including real hardware, can be simulated. Therefore, it is possible to verify the behavior of the GNC software in complete nominal and operative scenarios, in safe mode, the detumbling after release from launcher, and all the transitions between the different modes. Moreover, all the FDIR transitions can be tested if appropriate triggering of the events is possible between the OBC and the DKE. Indeed, the DKE can be required to include equipment complete failures or even signal conditioning features in order to perform error injections, therefore triggering data validation algorithms or higher level FDIR functions (see [Chapter 11](#) - FDIR development approaches in space systems).

It is well known that sometimes the interaction between the GNC software and the system software can be tricky: extensive documentation and proficient teams' involvement could be not enough to ensure the correct functioning of all the aspects in the most exotic, unlikely cases. Thus, it is of utmost importance to test the correct integration of the GNC software with the system software to ensure adequate working of all the functions in all the scenarios, and this can be easily done in this test set-up, since the flight software runs on the OBC, therefore involving the integrated version of the system software and GNC software.



In-orbit test

IOT involves the complete mission system, including the spacecraft flight model, with the aim to verify correct high-level, in-orbit functionalities, performances, and operations of the system. It is generally divided into:

- *Satellite commissioning*. To verify the functionality of the space segment, i.e., the platform and the payload. This includes the Launch and Early Operation (LEOP) phase, where nominal functionalities of all the sub-systems are checked; once the LEOP is concluded, also the redundant equipment functionalities are checked and the expected performance deeply verified.
- *Cal-Val*. To calibrate and validate the payload and its products, typically ending with a geometrical calibration, that for LEO Earth observation mission can be respect to ground targets (ground control points).
- End-to-end verification. To validate the system performance, i.e., including the ground segment. Therefore, in this phase, the complete system chain is verified, from user submission request, handling, planning of operations, acquisitions, downloading and processing up to the level of products required.

The GNC subsystem is mainly involved in the first part of the IOT but plays an important role also for the Cal-Val. During the satellite commissioning, all GNC equipment are verified again to assess their operability and performance in the real flight environment. GNC software should not experience major “environmental” differences from the tests performed on-ground with a GNC/AOCS SCOE, since from the software point of view, a simulated environment makes no difference to the real one (if the simulation is appropriate).

During the IOT stage, some high-level requirements are verified, that it was not possible to verify before: indeed, there are cases for which an appropriate ground testing is simply not possible, or sometimes too expensive, so appropriate analysis is performed to demonstrate the GNC design complies to the requirements, but actual, complete verification can be carried out only once the spacecraft is in orbit. It is the case, for example, of the evaluation of the pointing performance of the satellite, that is mostly a system performance verification rather than an AOCS one, but it is out of doubt that AOCS plays a fundamental role in achieving the final result through its complete design since the very beginning of a project, being attitude pointing performance one of the major design drivers obviously. Another typical case is orbit control performance: typically, a maximum deviation from the reference orbit is specified, or sometimes the maximum deviation of the ground track with respect to its nominal reference. Analyses are carried out throughout the development of the project to demonstrate the system is capable of achieving those performances, but no ground test is possible to actually verify the compliance to such a requirement. Therefore, the IOT involving all the orbit control chain is of utmost importance: from ground control to correctly propagate the real orbit of the satellite in order to plan the maneuver and to command firing start and duration (if not computed autonomously on-board), to the real firing of thrusters on-board the spacecraft impacting satellite dynamics, and to the attitude control during burns.

References

- [1] European Cooperation for Space Standardization (ECSS), ECSS system: glossary of terms, Technical Reports (2012). ECSS-S-ST-00-01C.
- [2] European Cooperation for Space Standardization (ECSS), Satellite attitude and orbit control system (AOCS) requirements, Technical Reports (2013). ECSS-E-ST-60-30C.
- [3] European Cooperation for Space Standardization (ECSS), Control performance guidelines, Technical Reports (2010). ECSS-E-HB-60-10A.

- [4] European Cooperation for Space Standardization (ECSS), ESA pointing error engineering handbook, Handbook, Technical Reports (2011). ESSB-HB-E-003.
- [5] A. PACKARD, J. DOYLE, The complex structured singular value, *Automatica* 29 (1) (1993) 71–109.
- [6] A. Megretski, A. Rantzer, System analysis via integral quadratic constraints, *IEEE Transactions on Automatic Control* 42 (6) (1997) 819–830.
- [7] C.L. Foster, M.L. Tinker, G.S. Nurre, W.A. Till, Solar-array-induced disturbance of the Hubble space telescope pointing system, *Journal of Spacecraft and Rockets* 32 (4) (1995) 634–644.
- [8] J.M. Hanson, B.B. Beard, Applying Monte Carlo simulation to launch vehicle design and requirements verification, *Journal of Spacecraft and Rockets* 49 (1) (2012) 136–144.
- [9] J.M. Hanson, B.B. Beard, NASA TP-2010-216447, 2010. http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100038453_2010042045.pdf. Accessed 04 Aug 2022.
- [10] European Cooperation for Space Standardization (ECSS), Software, Technical Reports (2009). ECSS-E-ST-40C.
- [11] <https://www.mathworks.com/help/sldrt/ug/tune-block-parameters-and-matlab-variables.html>.
- [12] <https://www.mathworks.com/products/embedded-coder.html>.
- [13] <https://www.mathworks.com/products/simulink-check.html>.
- [14] <https://www.mathworks.com/help/simulink/ug/select-and-run-model-advisor-checks.html>.
- [15] R. Bagnara, A. Bagnara, P.M. Hill, The MISRA C coding standard and its role in the development and analysis of safety-and security-critical embedded software, in: International Static Analysis Symposium, Springer, Cham, August 2018, pp. 5–23.
- [16] MathWorks Automotive Advisory Board Guide, Controller Style Guidelines for Production Intent Using MATLAB, Simulink and Stateflow, Version 1.00, MathWorks, April 2001.
- [17] <https://www.rtems.org/>.



On-board implementation

David Gonzalez-Arjona¹, Vincenzo Pesce², Andrea Colagrossi³,
Stefano Silvestrini³

¹GMV Aerospace & Defence, Madrid, Spain

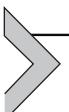
²Airbus D&S Advanced Studies, Toulouse, France

³Politecnico di Milano, Milan, Italy

This chapter presents an overview of the possible on-board architectures for guidance, navigation, and control (GNC) algorithms. First, the most important avionics alternatives for on-board spacecraft implementation are introduced. General processors, digital signal processors (DSPs), graphical processing unit (GPU), field programmable gate array (FPGA) and their historical background, application-specific integrated circuit (ASIC), and system-on-chip (SoC) are described, highlighting strengths and weaknesses of each alternative. A detailed discussion on alternative accommodations of the GNC functions into the on-board software (SW) is presented, making use of practical examples. Finally, different architectures for on-board implementation are discussed and the main verification techniques for on-board solutions are presented.

The content of this chapter is structured as follows:

- *Spacecraft avionics.* This section introduces the main alternatives for spacecraft avionics. Different components for on-board implementation are critically detailed.
- *On-board processing avionics.* In this section, the most relevant components, specific for on-board implementation are presented. Alternative accommodations of the GNC functions into the on-board SW are discussed.
- *On-board implementation alternatives.* This section summarizes and highlights the most relevant architectural alternatives beyond the central processing unit (CPU)-based approach, including hardware (HW) accelerators.
- *On-board implementation and verification.* In this section, the main steps to verify the on-board implementation are discussed. Profiling, technical budgets, and schedulability analysis are presented along with others verification steps.



Spacecraft avionics

As current and future space exploration missions are getting bolder, so do the requirements on the GNC algorithms and subsequently on their on-board implementation in the avionics modules and on the associated data interfaces. The avionics used in spacecraft and satellites are based on multiple subsystems, such as thermal control electronics, electrical power, mechanisms, telecommunication, attitude and orbit control, central processor, on-board data handling (DH), etc., all interconnected with on-board buses and networks. Placed within this high-level on-board system architecture, the “high-performance computing” subsystem will be of utmost importance for certain future space missions such as active debris removal, target bodies approach, descent and landing or rendezvous applications, just to name some. Such a subsystem must process in real time the inputs coming from the sensors of the spacecraft by utilizing dedicated “payload SW or HW” units. The future of on-board computers (OBCs) can benefit a lot by studying and developing high-performance solutions to support advanced and complex GNC of the spacecraft.

Considering the whole GNC chain, navigation is certainly the most demanding element in terms of computational resources. This is especially true if complex sensor modeling (e.g., image processing, light detection and ranging [LIDAR] processing) is involved. For this reason, the HW of navigation processor or coprocessor should be carefully selected. Different options exist and they may appear under different names, sometimes using proprietary nomenclature being developed by certain companies, but, in general, we can gather the processing devices in four big technologies: general purpose processors/microcontrollers, DSPs, GPUs, FPGA, or specific ad hoc electronic circuit. Among other technologies that will be partially covered by the above general classification, we might find other HW functions specialized such as video processing units, image signal processor, artificial intelligence (AI) engines, neural processing unit, programmable logic device, tensor processing unit, etc. These specialized options are not presented in detail in this chapter, but it is important to underline that they are feasible options that can be analyzed for specific coprocessing tasks. Furthermore, we will not cover either the processing functions that are not implemented as HW devices, such as emulators, simulators, virtual machines, interpreters, or other virtual processing functions. Fig. 13.1 summarizes the main HW processor/coprocessor options that are detailed in the following paragraphs.

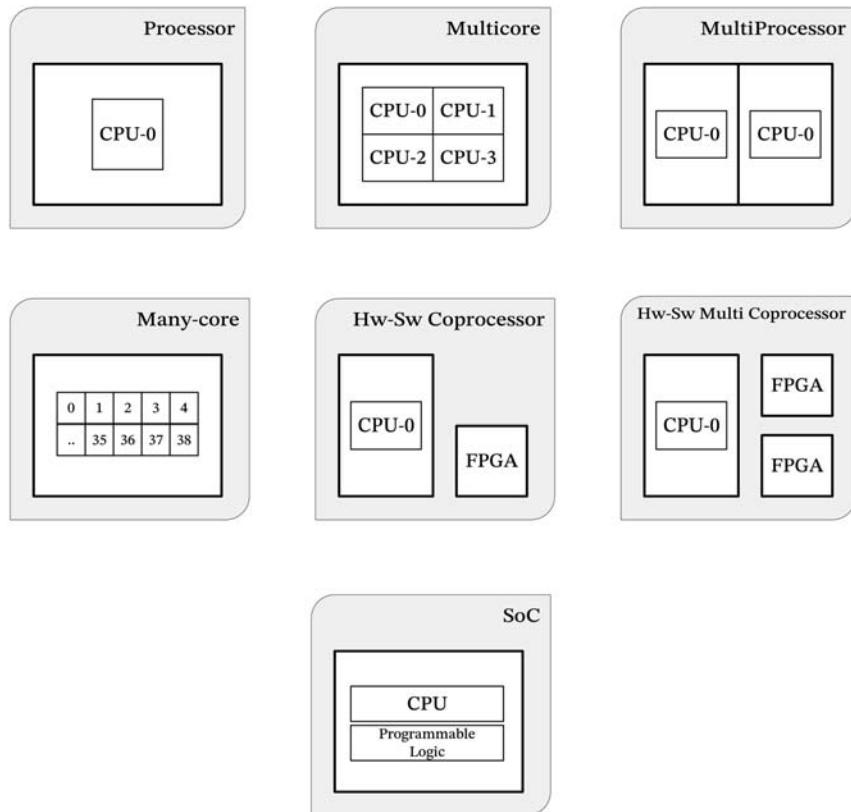


Figure 13.1 Different general on-board processing architecture options.

General-purpose processor or microcontroller

A processor is a predefined chip architecture that executes sequentially a program that is defined by SW sentences. It is the most common approach and is well known to the public, as they are core part of computers or mobile phones as one of the best examples. The processor HW includes arithmetic units, internal registers, buses, memory access, and different execution pipeline steps fixed and standardized for each defined processor architecture and version. The engineer programs the SW sentences using a fixed and standard predefined set of instructions. Different SW abstraction languages can be used to create the programs to be executed in the HW processor. An SW compiler is used to translate higher-level SW languages into the low-level set of instructions accepted by the processor/microcontroller HW architecture pipeline. An operating system might be used to govern the processor

behavior and configure the different peripheral HW that might connect to the processor to extend functions or memory capabilities. It is not needed to count on an operating system though to execute a program. When no operating system is used, we refer to bare metal application SW. An operating system might be general purpose or specific, and among some of the specific versions, it is very important to highlight the real-time operating systems that are used for critical operations such as the GNC functions for a spacecraft, satellite, or, back on Earth, for automotive. A real-time operating system will ensure that different tasks of the SW to execute, being part of the same application or different ones, will always consume a known amount of time or the task will be canceled if the time is exceeded. This is done to manage other tasks which criticality (or due to engineer decision) might not wait. The architecture of a processor defines the number of execution steps of the HW units, the communication buses, the data-word widths, and the memory accesses among others. We might find multiprocessor and multicore versions. A dual-core, quad-core or generally multicore options are those that incorporates more than one identical HW central processing block in the processor. Each core will nominally count on their specific low-level random access memory (RAM) level 1 (with the level defining memory hierarchy), dedicated to the core, but may share RAM level 2 or 3 with the rest of the cores. A multiprocessor, as the name indicates, consists of different processors (multicore or unicore) interconnected by communication buses and sharing some higher-level resources such as peripherals of memory RAM of level 2, 3, or beyond. The general-purpose processor counts with powerful and precise floating-point operation units for complex mathematical operations at high-bit resolution. The general-purpose processor or microcontroller unit (MCU) is usually the “brain” core part of the mission on-board system (on-board SW [OBSW], DH, command and control) and will execute GNC or attitude and orbit control system (AOCS) as an application software.

Digital signal processor

DSP is a microprocessor specifically designed to process digital signals. Digital signal applications require real-time data processing over an input stream of digital values (or previously digitalized from an analog signal). The data nature is based on signals such as video, audio, images, or global navigation satellite system signals, which may require complex or time-consuming mathematical operations. A DSP, being more specific than a general-

purpose processor allows some of these complex mathematical operations in one clock cycle, and, as an example, it is very common to execute multiply-add-accumulate operations, in one clock cycle allowing much faster execution times than a normal processor. DSP can be used for mathematical operations without a physical digital signal being processed but just an algorithmic solution. However, most of the processing advantage is gained, thanks to the processing in streaming. Therefore, if the functions include many branch-decision statement, the DSP is not really providing any acceleration advantage and will in fact not facilitate function control in the same way a processor can do. The DSP can be used as main processor of the avionics architecture, but it would eventually be more devoted as a coprocessor implementing functions such as convolutions, correlations, filters, downsamplings, or discrete transformations. A DSP board may include more than one DSP core, interconnected by means of function control logic and memory accesses, like the network-on-chip architecture. To program the DSP functions, a dedicated library of functions in SW is used, potentially as an extension of known processor SW languages such as C.

Graphical processing unit

GPU is another microprocessor specialization, in this case fully devoted to image and video processing, providing multiple processing elements (arithmetic-logic unit) managed by an execution unit and connecting to memory blocks independently and with the capacity to be parallelizable. The programming of GPU nodes execution is made in specific SW language (CUDA is a well-known example) and combines set of data plus instruction in the same statement or execution thread rather than a separation of processing instruction that will use data from other sources. This is one of the programming differences versus a general-purpose processor. Counting with multiple processing units and data channels, it allows complex operations over vast amount of data as it is a video engine in a very fast response compared to normal processors. The drawback comes on the power consumption that is also much higher than general processors.

Field programmable gate array

FPGA is an embedded chip, a digital processing component that provides an HW architecture that can be programmed to describe a specific HW component. An FPGA is a programmable and reconfigurable HW device for implementing digital systems, from low-level components to complete

SoC and board-level designs. The FPGA is an HW matrix of logic gates implementing different functions that can be interconnected selecting different routes that are programmable. An FPGA developer will: “*Program the physical HW of the device rather than write SW to run on a predefined HW (processor, DSP, or another device).*” When programming an FPGA, you have to abandon all preconceived notion of a sequential SW and think in terms of a digital and parallelizable paradigm. As already described, general-purpose processors, embedded processors, GPUs, and DSPs are a fixed HW component where a dedicated SW is programmed and executed in the HW, following a set of rules to exploit the available HW and the fixed functionality it can provide. In an FPGA, the HW component itself is programmed, as well as the digital functions you want your HW will have, and the interconnections and processing rules are created. In other words, a general-purpose processor can be programmed with the desired functions and rules. Similarly, a GPU or a DSP can be programmed, and you can also create an HW component in the FPGA that afterward can execute SW created by others. FPGAs are sometimes also used in the development process of device prototype. This is done before manufacturing the final fixed design to create an ASIC of a processor component. Some FPGA technology can be reprogrammed many times while an ASIC is fixed and cannot. The different FPGA technologies include flash-based FPGAs, antifuse, and SRAM-based FPGAs. Antifuse FPGAs are one-time programmable devices. SRAM-based are fully reprogrammable and flash-based might be reprogrammable up to a number of times. Flash-based FPGAs are nonvolatile devices that retain the programmed HW functionality even if reset or power cycled, while for SRAM-based FPGA, the configured functionality should be programmed each time a reset or power-off/on happens as it is based on volatile technology. To program FPGAs, it is used a hardware description language (HDL), which resembles to a SW but with key differences. In fact, an HDL describes HW using SW-like coding, but FPGA programming involves architecture definition, functionality definition, and physical constraints, including timing, routing, input/output interfaces, pin-out of the chip to specific parts of the defined function, interconnection of embedded HW blocks in the FPGA logic, and even voltage-level decisions or pull-up/pull-down configuration of pins. The FPGA programming allows implementing different functions that will work in parallel, either if they interact among them or not. The functions can be implemented in a pipeline of interconnected parallel partial executions of functions. The FPGA is normally used as coprocessors to accelerate certain functions or to implement

fault-tolerant softcore general-purpose processors in their logic such as Leon processors, RISC-V, Microblaze, or NIOS to name some.

FPGAs in space: history, present, and future

A review of the utilization of FPGA in space segment is crucial to understand the flight heritage and the target applications in the past, in the present, and the evolution toward the future. One of the most adopted FPGA technologies in space satellites and missions is the FPGA by Microsemi. Their anti-fuse technology (as well as previous other FPGA on-time programmable solutions) allowed for a technological step forward in the utilization of FPGAs in space. Microsemi [1] FPGA portfolio for space includes antifuse RTXS-SU and RTAX series, flash-based rad-tol RT ProASIC3, and the larger device RTG4 and has a new product for high performance such as the RT PolarFire. RTSX-SU has flight heritage since 2005 in missions such as the Mars Reconnaissance Orbiter. RTAX are widely used in many missions, with flight heritage since 2007 in missions such as the Mars Science Lab Curiosity rover. RT-ProASIC3 has flight heritage since 2013, being the first flash-based RT FPGA in space, in missions such as NASA IRIS. Other missions using RTSX-SU include GPS 2R-M Program, New Horizons, SAR-Lupe 1 and 2, Galileo GIOVE-A, and TerrSar X. RTAX-S is on-board COSMO SkyMed 1 and Mars Phoenix.

Many FPGAs are used for mass memory controller and implementation of interfaces protocols with a good example of the many implementations of SpaceWire links and SpaceWire routers in FPGA [2]. Exomars 2022 OBC1 [3] includes two (four due to redundancy) RTAX2000S for Interface, Reconfiguration, and Memory Module, including Mass Memory control. The ESA-OBC1 manages the whole ExoMars 2022 mission during Cruise, EDL, and Mars Surface Operation phases running the whole Mission Software [4]. Utilization of Microsemi for relatively large capacity and high-performance needs involves the utilization of the latest devices such as the RTG4 that is a flash-based FPGA with reconfiguration capabilities up to a certain number of cycles (~ 300). Being relatively new component, there is not enough information on flight heritage at the time of the elaboration of this book. It is being incorporated though in different on-going missions such as in ESA mission HERA being interfaces and spacewire router controller of the main OBC avionics or in the RVS3000-3D [5–9] pose estimation high-performance in-orbit computing platform that includes LIDAR and image processing into a single box including the RTG4 as

the main processing device to cope with LIDAR data extraction and algorithms execution using it. This single box is a solution for real-time calculation of 6DOF information with the application of iterative closest point algorithm and including matching between LIDAR scans and target CAD model.

Xilinx devices provide an alternative for space usage, implementing SRAM-based FPGA technology. In the Venus Express Monitoring Camera (VMC) and the Dawn Framing Camera [6], the Xilinx FPGAs were used for image-data compression. Virtex4QV is used for Data Processing Unit (DPU) functions such as the implementation of a LEON2 processor core running the RTEMS RTOS and application-specific coprocessor for image processing. The Framing Camera DPU is based on a similar FPGA-based DPU designed for the ESA's VMC. In the ESA's Solar Orbiter, multiprocessor architecture is implemented for high-performance floating-point computations in the PHI instrument [7] for scientific analysis using single instruction, multiple data (SIMD) architecture as an accelerator within its DPU. This DPU is based on LEON processor and two Xilinx XQR4VSX55 to execute RTE inversion.

Xilinx FPGAs have leveraged the FPGA utilization from performing a predominantly flight ASIC prototyping role to being designed into and flown in projects like the Mars lander and rovers [10]. Xilinx rad-tol or rad-hard (RH) FPGA based on SRAM technology might be more susceptible to single-event upsets (SEUs) but are the key devices to include HW-acceleration capabilities for high-performance applications to cope with high frequency of operation requirements.

Recently, in the Mars2020 rover (Perseverance), NASA includes HW-accelerator board in the Vision Compute Element [11], including the Virtex5-QV as the Computer Vision Accelerator Card to aid in landing navigation and autonomous driving on the Martian surface executing acceleration part implementing certain stereo and visual tasks such as image rectification, filtering, detection, and matching. It is necessary for the core execution of the Terrain-Relative Navigation and the Lander Vision System comparing on-board pregenerated map with the captured images of the landing site during descent to compute relative position of rover to ground.

In the latest years, there is a new vendor that is fully dedicated to the development and commercialization of space RH FPGA in Europe, the company NanoXplore from France. Their portfolio offers SRAM-based fully protected RH FPGA such as NG-MEDIUM, NG-LARGE, NG-ULTRA, and ULTRA300, being the NG-ULTRA a SoC product

embedding the European DAHLIA processor based on quad-core ARM. This is a very new technology with only some components qualified, as the case of the NG-MEDIUM and the on-going NG-LARGE and with yet no flight heritage.

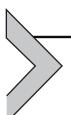
New SRAM-based FPGAs in the market are not always being developed as a RH technology product by design but as a radiation-tolerance device taking advantage of the new fabric-nodes capabilities. Therefore, there is a potential path to flight for commercial FPGA devices that offers future missions a significant boost in capability compared to existing systems. Kintex Ultrascale+, and, in particular, the KU060 is one of those rad-tolerant devices examples which are being included for different new processing requirements in space domain as a coprocessor suitable for implementing different high-speed communication protocols based on optical links or deployment of complex in-flight deep learning solutions as in Ref. [12]. These kinds of devices pave the way in between a full RH chip and the risk of using a commercial one, as no risk would seem considered for critical operations. More examples for this kind of “New Space” approach for on-boarding FPGAs which are not fully or none RH protected can be found in Ref. [13] where missions using CubeSat, mostly for low Earth orbit (LEO) noncritical operations, benefit from the utilization of SoC FPGA-based computers which are not available yet in RH technologies.

Application-specific integrated circuit

ASIC is a dedicated HW design that, using building blocks or HW submodules, reproduces a particular HW function in an integrated semiconductor. The final outcome is a specific HW chip. A general-purpose processor is, in reality, an ASIC. The goal of ASIC production is the creation of the processor chip itself and not the programming of a function that would execute in that processor later. The ASIC does not exist before the HW functions we want to implement are designed, being more specific than an FPGA. The FPGA HW itself exists, it provides matrix of interconnectable logic gates, and we design and program how to use the FPGA general HW to create specific HW functions. In the case of an ASIC, there is no programmable HW because there is, in fact, no HW. The engineers define what that HW will be (e.g., a processor, an FPGA, a DSP, or power control regulator). Understanding the application-specific concept, the reader must then realize also another big difference with FPGA technologies which is that FPGA might be reprogrammable, and ASIC will not.

System-on-chip

In the last years, the combined use of SoC-integrated HW processing solutions is gaining more and more relevance. SoC combines most of the advantages described above in the other processing options by integrating multiple HW parts into a single device (e.g., Processor plus FPGA or DSP). Simply integrating the processor and FPGA components into a single SoC FPGA potentially reduces system power by 10% to 30%. The Inputs/Outputs (I/Os) that carry the signals between devices, often at higher voltages, are one of the most power-hungry functions in an application. Therefore, the SoC architecture provides fast intrachip communication between all HW components and low power/size. By relying on tightly coupled accelerators, peripherals, and processors, the engineers can develop efficient HW/SW coprocessing solutions for a variety of signal processing and GNC techniques. This SoC solution increases the flexibility during implementation, but the HW production into space format is more challenging than separated options. In principle, the SoC architecture can support all kinds of accelerators, including FPGA, DSP, GPU, or even multicore. Overall, implementing sophisticated GNC (moreover if involves AI or computer vision algorithms) on embedded platforms, regardless of their FPGA- or GPU- or DSP-based HW, is a very challenging task. Porting or designing the entire algorithm on an accelerator (especially FPGA or GPU) is neither efficient nor feasible (in a reasonable amount of development time). In particular, a small part of the algorithm pertains to infrequent high-level decision/control operations and complicated sequential functions, which are not worth accelerating. These parts are better to be executed on a low-power general purpose processor, which moreover, increases the flexibility of the design both at run-time and design-time. Therefore, the potential to combine the strengths of dedicated HW accelerators with flexible general-purpose processor is the main reason of today's trend to use embedded SoC technologies.



On-board processing avionics

Electronic components and spacecraft avionics need special protection when exposed to the harsh space environment. In particular, processors for space are required to be tolerant to radiation and environmental effects such as total ionizing dose (TID), latch-up events, SEU, single-event transient, single-event functional interrupt, and temperature cycles. As an example,

tolerance to radiation TID should be 100–300 kRad for geostationary orbit and beyond and 10–50 kRad for some LEO mission. Another reliability issue is the complete and permanent failure of a processor or a critical sub-component. The avionics processing shall deal with temporal upset or failures, avoid propagation of error to other equipment, modules, or system, and prevent from permanent damages or failures. The problem is that most of the available avionics processing devices and integrated systems-on-chip cannot offer high tolerance to environmental conditions (radiation effects, temperature ranges, pressure, vibration ...) and offer high-performance capabilities at the same time. The space environment and the mission concepts impose restrictions or difficulties to design an avionics architecture or select a processing component to make extensive data processing tasks on-board of a spacecraft considering technology that withstand the mission's environment (temperature, vacuum, radiation, and fault-tolerance). Reliability is a must not only for operations that are critical or risky but also considering catastrophic failure that would imply the loss of the component. Contrary to Earth applications (automotive, trains, data centers, etc.), the component launched into space will, in the majority of the cases, not be repaired or replaced. Therefore, for all these reasons, the space-qualified portfolio of high-performance processing solutions is quite limited, considering real-time critical operations for GNC, DH, OBSW, communications or for sensor data processing in flight, also including data compression. Critical autonomous on-board navigation concepts based on navigation camera images are one of those scenarios that may require considerable on-board avionics processing capabilities. Those solutions, targeting different navigation scenarios such as descent and landing into small bodies, satellite servicing, or rendezvous operations, require architectures and processing technology above the standard space-qualified processors. The standard for current missions in Europe is to adopt fault-tolerant space processors such as the dual-core LEON3 or the quad-core LEON4 for SW solutions. These processors represent already a big advantage compared to previously available processors, but they still might require additional and dedicated co-processors based on large high-performance devices such as dedicated DSPs, GPUs, SRAM-based FPGAs, or even dedicated ASIC to perform digital signal processing over acquired images.

In addition to environmental constraints, power consumption is a limiting factor in space, targeting on-board processors consuming 10–20 Watts, or up to 50 Watts at the very limit. This restricts also power-hungry solutions unless strictly necessary. The main design objective is to

select processing avionics technology offering a good trade-off of performance per Watt with adequate processing architecture.

Historically, the majority of space missions have relied on rad-tolerant and RH IBM PowerPC and Gaisler LEON CPUs. They can be implemented either as softcores (e.g., in an FPGA) or as hard intellectual property on ASIC. Due to their subpar processing speed, softcores can only serve as a secondary solution for HW/SW codesign.

Space processors can be divided into commercial off-the-shelf ruggedized, fabricated on dedicated RH processes, and the ones that are radiation hardened by design (RHBD), i.e., RH is achieved by design techniques in the layout, circuit, logic, and architecture areas, designed as ASIC and fabricated on commercial complementary metal oxide semiconductor (CMOS) processes. Most RHBD processors are based on the success for European LEON processor architecture. During the last years, the evolution is drastically oriented toward providing better performances, including other architectures such as ARM or RISC-V processors. Lately, one of the most interesting concepts is the use of SoC or heterogeneous processing platforms including different processing technologies such as multicore processors combined with FPGA, GPU, or dedicated HW engines floating-point units (FPUs), SIMD, or even AI-dedicated blocks.

New concepts and new versions of onboard processors for space application are continuously being developed, and only the most important ones are collected into a not exhaustive list that, however, will give the reader information about producers and specific products:

- *SPARC V7 ERC32 and TSC695Fl.* Made by Atmel in France, originally a three-chip set, it is now a single chip CPU. It has been used in many satellites and systems in Europe and elsewhere. It achieves 12 MIPS/6 MFLOPS at very low power (0.3W for the core excluding I/O). It has been used by several companies to develop their dedicated single-board computers.
- *Atmel LEON2 AT697.* It provides a major step forward from the SPARC V7 processor. It is a LEON2 SPARC V8 processor based on IP core provided by Aeroflex Gaisler. The chip includes the processor, memory interface, and a PCI interface and executes at 70–80 MHz. Several LEON3-based SoC have been developed recently by different companies and with different design architectures.
- *LEON4.* It can be utilized in both asymmetric multiprocessing and symmetric multiprocessing (SMP) configurations. A typical four-processor system implemented as a 65 nm ASIC is targeted to be capable of

delivering up to 4300 Dhrystone MIPS at 800 MHz. The processor provides HW support for cache coherency, processor enumeration, and SMP interrupt steering.

Accommodation of GNC functions into the on-board SW

The avionics architecture distinguishes different SW process agents in the accommodation of the GNC system: central OBSW, DH SW, GNC or AOCS functions, sensor management, Telemetry and Telecommand (TMTC), and security functions. In terms of HW avionics trade-off, one first architectural decision is on the deployment of the GNC functions, combined into the instantiation of OBSW and Navigation (GNC) in different processors or in one unique single processor with both functionalities.

There are several deployment options, but the trade-off is restricted at functional level on the following two options as in Fig. 13.2.

- *Option A.* OBC SW and DH in dedicated processor and GNC offloaded in a coprocessor. The selection of the instantiation of the OBC tasks and system DH is running on a dedicated processor that might be simpler while the GNC runs in a dedicated coprocessor device with better performances.
- *Option B.* OBC, DH, and GNC are deployed into a unique processing system.

The two architectures offer advantages and drawbacks, and the choice must consider the mission needs. In fact, with only one processor as in Option B, there is a risk of overloading the processor with all the SW tasks, and due to the high criticality of OBC tasks, there is a risk of not isolating GNC functions so to lose execution step processes, or image frames in case of

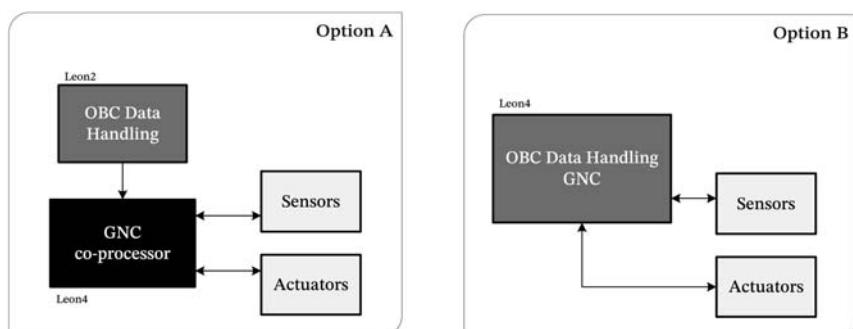


Figure 13.2 Example of processors identification for OBC and GNC functions.

optical navigation. The Option B could count on a powerful single-core or multicore processor gathering all the platform SW functionalities including OBC, DH, and GNC. Moreover, it can simplify architectural number of elements, considering that the SW tasks need CPU load enough for both OBC and GNC executions. The problematic part still remains in the isolation of critical functions. A safe solution is to isolate the OBC and DH function from the GNC concept, as in Option A. For many missions though, the performances and CPU load may not be very exigent, and it is not necessary a powerful processor to reach the performance requirements for the OBC and DH functions. Navigation coprocessor performance requirements are relaxed in Option A having a dedicated HW.

This is the very general and very first avionics architecture decision, afterward, it is needed to explore in more detail the options for a GNC/Navigation coprocessor trade-off and which avionics technology to use. At system level, the trade-off for the GNC coprocessor might be based on the combination of different processing technologies as the ones presented at the beginning of this chapter, processors, DSP, GPU, ASIC, and FPGA.

In the architectural trade-off at technology level, we may concur then that a good candidate avionics processing solution would be based either in a microprocessor or a combination of a processor and an HW accelerator. The trade-off at system level for architecture of the GNC processing avionics would then be based on three configurations as in Fig. 13.3:

- *Architecture A.* Pure SW solution in space-grade powerful processor
- *Architecture B.* HW/SW solution in space-grade powerful processor and high-performance HW accelerator
- *Architecture C.* HW/SW solution integrated chip including space-grade HW accelerator + processor embedded hard IPs

Let's analyze this example as a potential exercise for the architectural decisions for the GNC accommodation into a specific on-board implementation to satisfy the mission requirements in terms of real-time capabilities, operation deadlines, and important technical budgets such as power consumption:

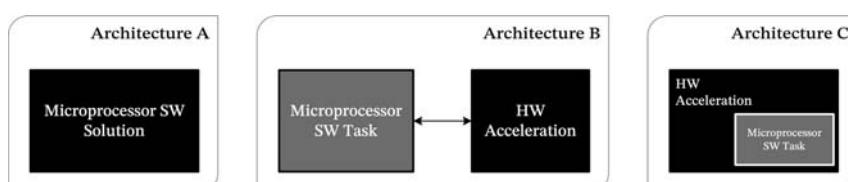


Figure 13.3 Example of system-level architectures.

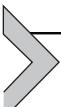
Architecture A. The Pure SW solution in space-grade powerful processor has the advantages of including only one device which in terms of power consumption is potentially consuming less energy than two devices of equivalent power consumption characteristics. Besides, there is no partitioning of task between devices, and hence there is no intercommunication needed. The disadvantages are that the CPU load will be very high, close to the 100%, in order to satisfy GNC performances and considering the available space-grade devices the total system frequency will be lower than a solution based on a coprocessor offloading consuming tasks. On top of the performances constrains, if only utilizing one processor, this means that all the interfaces with the rest of the system will have to be managed in the processor at the same time of the algorithms execution, and therefore there is less available CPU load for the algorithms.

Architecture B. HW/SW solution in space-grade powerful processor and high-performance HW accelerator improves the performances of the pure SW solution as it offloads algorithm consuming tasks into a dedicated device. HW acceleration of a coprocessor device is utilized to achieve high-frequency performances. Moreover, the HW accelerator coprocessor provides programmable routing which allows flexibility for implementing all the needed interfaces, communication, and arbiter of data without impacting in the system performance (again offloading the communication part from the SW processor). In terms of fault-tolerance, the HW accelerator will be devoted to specific tasks (e.g., image processing) which are time consuming. Being instantiated in a different device with respect to the navigation filter and other GNC functions, this architecture provides an automatic isolation of functionalities to protect propagation of subsystem errors into the GNC chain. The main drawback of this configuration is that power consumption increases compared to the full SW solution as there are now two elements instead of one.

Finally, the third option is the *Architecture C.* HW/SW solution integrated chip including space-grade HW accelerator + processor embedded hard IPs. This solution provides the best performances as in the Architecture B with low power consumption and data interchange as in Architecture A. This solution gets the good points from the two previous options. There is not yet any space-grade SoC qualified, but there are some in the near future roadmap considering FPGA accelerators (considering book writing time), the NG-ULTRA from NanoXplore or the radiation-tolerant RFSoC, Zynq, or Versal ACAP from Xilinx. The fact of having only one device is translated into less power consumption than having two separated devices

(of similar power consumption footprint) and less data exchange (that supports again less power consumption). There is still the possibility of HW acceleration to achieve high-frequency performance and offload SW processor, reduce data management, isolate functions, and provide flexibility for interfaces.

Besides studying and developing high-performance subsystems, an important challenge is to connect the new module to the remaining components of the avionics architecture. In this direction, the **S**pace **A**vionics **O**pen **I**nterface **a**Rchitecture (SAVOIR) initiative studies and analyses the architecture of space avionics in order to federate the space community to cooperate and improve the way that they build avionics subsystems. The output of SAVOIR is a standard for developing avionics and interfaces for different space missions by considering them as reuseable products and by adapting to a development-oriented philosophy. It translates into a more competitive framework for the European community with increased cost-efficiency and functionality for on-board building blocks.



On-board implementation alternatives

As we have already seen, some architectural alternatives can be identified beyond the CPU-based approach. In fact, modern advanced image processing systems and complex GNC algorithms mandate the design of new generation space avionics to include HW accelerators, e.g., FPGAs, GPUs, or multicore very long instruction word (VLIW) DSP processors. On-board implementation alternatives shall look for the proper architecture selection based on certain assumptions considering previous similar implementations, state-of-the-art for processing architectures, and mission restrictions. The trade-off analysis shall rate different quantitative and qualitative parameters such as:

- Development cost/time.
- HW cost.
- HW transistor area.
- Overall execution time.
- Overall GNC performances.
- Flexibility and Scalability.
- System mass and power.
- Future roadmap.

The first attempt to address the architecture alternatives investigation regarding high level avionics architecture is to combine different HW

components to grant performances, reliability and functional mapping. In the previous section, three alternatives were identified considering a generic HW accelerators. In this section, different architectures alternatives are detailed considering different HW acceleration alternatives.

Multiple processors

This approach ([Fig. 13.4 – left](#)) relies mostly on SW parallelization to accelerate performance. Communication between processors could be implemented using message passing or shared memory libraries depending on the specifics of the interconnection network. Generally, performance-wise, an architecture based on few processors cannot compete with platforms that make use of HW-based optimization/customization. The performance might be limited due to synchronization delays among processors and due to inherent inefficiencies of the general-purpose programming model of each processor. Moreover, it is very important to note the significantly lower performance of the space-grade processors compared to commercial ones. As an example, a profiled specific image processing C/C++ implementations of feature extraction and tracking on a Leon2 system (50 MHz space-grade processor) using 8-bit 1024×1024 pixel images shows that the performance obtained is $\sim 80x$ lower than a commercial Intel Pentium 4 at 3.00 GHz (80–150 s vs. 1–2 s per image, the elapsed time also depends on the utilized FPU).

Processor and multiple DSPs

The performance of the general-purpose processor can be enhanced by coupling it to a number of DSP cores as in [Fig. 13.4 – right](#). Due to their SIMD and customized instruction set architecture, the DSP cores are more efficient than the CPU in digital signal processing, e.g., convolutions on image, in a stream data flow. Generally speaking, if the algorithm to be optimized requires a sample rate below few kilohertz, then a single-channel implementation with a DSP might be a good choice. However, as sample

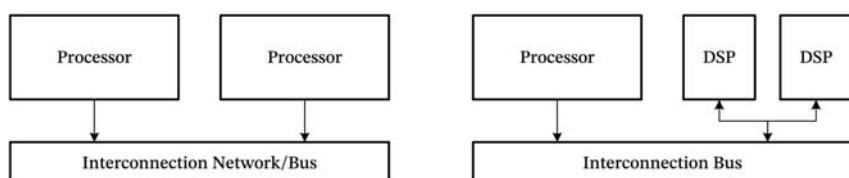


Figure 13.4 Multiprocessor (left) and processor with multi-DPS (right) architecture.

rates increase beyond a couple of megahertz, or if the system requires more than a single channel, or the number of DSPs should be proportionally increased, then such a system might struggle to capture, process, and output the data due to the many shared resources, buses, and even delays by the central processor itself.

FPGA

To overcome the performance limitations described above for processors and DSPs, we can make use of distinguished computational models, like the FPGA, which is intrinsically even more parallel than the previous solutions. Several possible architectures exist and some of them are presented here (Fig. 13.5).

Single FPGA

A single FPGA architecture can be employed. In this way, by exploiting a variety of parallelization and HW design techniques, we can significantly accelerate the intensive tasks (e.g., image processing algorithms). On the downside, sequential parts of the algorithm will perform the same or worse on FPGA than running on a processor. Moreover, we note that the available FPGA resources are limited, and the floating-point operations might not be the best fit for FPGAs (in general, the GNC system relies on a certain number of floating-point operations to achieve the required accuracy). Given these limitations and the increased development time on FPGAs, building a complete system might be done more efficiently when combining the FPGA and the CPU capabilities (parts of the algorithms running on a serial processor, others designed on FPGA depending on their amenability to HW acceleration). This solution is particularly suitable if the full system fits in a single FPGA chip (i.e., CPU plus FPGA logic tightly coupled in the same chip).

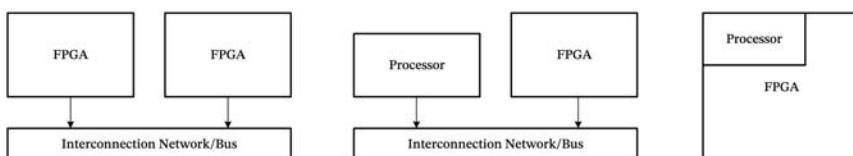


Figure 13.5 FPGA-based architectures.

Multi-FPGA

This multi-FPGA design (i.e., 2-FPGAs) is a simple solution to overcome the limited resources of an FPGA device. Two devices provide more resources to map bigger systems including more complex functions. On the downside, the interconnection between boards or devices creates new problems; the speed and reliability of the communication requires considerable design time, while the cost, size, and power consumption increase (compared to single-FPGA).

FPGA and hard-IP processor

In this design, we assume two devices, one FPGA and one processor. The partitioning of the system into SW-based functions (e.g., main flow control) and FPGA-based functions (e.g., parallelized to achieve faster execution) according to multiple design criteria (coming from the specifics of the HW and the requirements of the application) can bring very good results both in terms of performance and development cost/time. Data transfer between HW modules and processor is a very challenging task in this approach.

FPGA including softcore processor

With this solution, where the CPU is implemented as a softcore by using the FPGA logic, we combine the advantages of a single-device architecture and the advantages of the CPU-FPGA coprocessing. The main drawback is that a softcore provides considerably lower performance than an embedded hard processor (or an ASIC implementation). Additionally, the softcore processor utilizes a lot of FPGA resources, especially for its FPU, and hence, we limit even more the space available for implementing accelerators.

FPGA and FPGA including softcore processor

With this double-device approach, we essentially use two FPGAs, one dedicated to accelerators and one to the soft-IP processor. It offers more resources for the HW accelerators; however, it inherits the disadvantages of the double-FPGA solution described above and the low-performance of the softcore.

System-on-chip

One of the most promising architectures is represented by the SoC. As already discussed, it has the advantages of multiple HW architectures but with fast communications between HW components and low power and size.

FPGA SoC

A SoC including FPGA integrates the SW programmability of a processor with the HW programmability of an FPGA. Conceptually, the chip is divided between two areas: the “Programmable/FPGA Logic/Fabric,” which is equivalent to a conventional FPGA, and the “Processor/Microcontroller System,” which includes the CPU and the peripheral units. The hard IP processor inside the SoC is surrounded by hard IP peripherals, which can provide low-power and fast-access to the external RAM, storage devices, communication I/O, and most importantly, to the FPGA logic of the chip. On the FPGA side, we can implement VHDL/Verilog accelerators to speed-up intensive algorithms. Hence, the FPGA SoC allows for flexible and scalable designs with reduced power dissipation and increased performance.

DSP or GPU SoCs

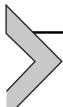
An SoC can also be constituted by a GPU or DSP, and, in this way, it integrates the SW programmability of the CPU with the more specialized SW programmability of a DSP VLIW processor or a GPU. More precisely, the CPU and its peripherals are connected to a number of “DSP cores” (1 to 8) or smaller “CUDA cores” (e.g., up to 256), all placed within the same chip. The cores take advantage of their SIMD and/or VLIW architecture to accelerate the execution of repetitive instructions like those issued when processing 1D, 2D, or higher dimensionality signals. The CPU and the acceleration cores (DSP or GPU) are connected to a number of peripherals (very similar structure to the one mentioned above for the FPGA SoC), which include hard IPs for I/O, or even for accelerating common functions (e.g., image compression) to boost the I/O of the chip at relatively low-power and high-speed performance. The DSP- and GPU-based SoC rely on C/C++ language to program, which is generally considered easier/faster than the VHDL programming of an FPGA, however, at the cost of increased power-performance ratio.

ASIC

ASIC is a dedicated HW design reproducing a particular HW function in an integrated semiconductor. The main advantages of ASICs versus FPGAs are speed and power consumption. ASICs offer more opportunities for optimizations on speed and low power consumption techniques. ASICs also allow implementation of analogue circuits but at the price of high nonrecurring engineering cost and lack of designing flexibility (increased development cost, both for SW tools and manufacturing, decreased flexibility in redesign and implementation phases).

ARM-based ASIC CPU

Arm-based ASIC CPU can accommodate FPGAs, and this architectural solution can provide more than 1000 MIPS even with single-core execution. However, they are still one order of magnitude slower than architectures including one or more high-performance HW accelerators. Therefore, in general, high-performance avionics architectures must be designed using HW accelerators.



On-board implementation and verification

Independently from the on-board avionics HW selected, the actual on-board implementation shall be based on the SW concept design or the HW/SW codesign of the different functions of the GNC. These functions are integrated into the spacecraft (platform bus controller) infrastructure components considering the OBC, the existing coprocessors, the mission control system, DH, sensor management functions, and ground station interfacing.

HW/SW codesign is based on concurrent development approach for both HW and SW functions at the same time. One of the most important tasks is the partitioning of the system functions in order to assess the fulfilment of requirements by HW acceleration while maintaining critical decision-making control in SW.

The programming and documentation standards applicable to the project shall be followed during the implementation production phase, starting from a first review, analysis, and tailoring when needed. Commonly in Europe, under ESA programs, the applicable standards are the European Cooperation for Space Standardization (ECSS).

The implementation and verification process will depend on the criticality level evaluated for the subsystem and mission. The ECSS define, for instance, the SW criticality category A, B, C, or D based on whether a failure in the module to develop may cause a human death, mission failure, partial mission failure, system or subsystem failure, and what is the impact of a non-recoverable failure in the SW if it means losing the mission purposes. Highest criticalities A or B will involve extra verification and quality assurance/product assurance activities. One example for high-criticality SW is the implementation of an independent SW validation and verification campaign to be performed by a third-party company different than the developer one.

The on-board implementation shall also encompass the following design decisions beyond the pure GNC functionality and interfaces:

- Identify target avionics processing HW for the mission.
- Identify representative developments boards equivalent to the final selected HW.
- Analysis on the need of processor plus coprocessor architectures (complex functions offload to coprocessor, main functions remain in processor).
- Identification of coding language (in processor and coprocessor).
- Identification of operating system (commonly real-time operating system).
- Analysis on the need of SW isolation (TSP techniques for time-space partitions, hypervisor).
- Schedulability analysis (frequency and latencies for each defined function tasks, including execution time deadlines).
- Mathematical libraries or other third-party modules to be used (qualified already is preferred).
- Technical budgets (CPU-load, memory utilization, HW programmable resources, power consumption).
- Clocking scheme and synchronization of elements (on-board time propagation).

One of the first implementation tasks for on-board applications is the feasibility analysis and design of real-time system. The GNC might be already designed and even a prototype is implemented (for instance, using mathematical programming languages) defining the needed functionality, but from that prototype till the current on-board implementation, several adaptation steps are needed. The final flight SW shall work in a real environment, considering real HW timing, delays, interlocks, data resources access, in accordance with the true time (actual time) of real-world events, dynamics, and kinematics. It shall also operate to acquire sensor values and to command actuators to control the spacecraft in the specified and needed response time.

The implementation onto the target HW platform shall include pure GNC algorithmic functions and additional functions such as sensors management, AOCS, TMTTC dedicated manager to interface GNC functions, state machine with operational and nonoperational mission modes for the GNC SW, data storage manager, interfaces definition with other SW functions or OBSD or external HW units, fault detection, isolation, and recovery (FDIR) function or support to main system FDIR (providing flags or measurements) and in some cases specific sensor data processing functions. One example of dedicated sensor data processing function is, for optical

navigation solutions, represented by computer-vision algorithms that extract navigation features from images. This is a nonnegligible complex task that may require a dedicated SW task and predefined rates, or execution separated from main pure GNC navigation filter. Not all these functions may be present when developing a mathematical model of the GNC functionality in tools like Simulink or Matlab but shall be present in the final implementation onto the selected processing avionics, and they should be properly designed and orchestrated.

A first SW version of the GNC solution can be prototyped and used in the SIL test campaign (see [Chapter 12](#) – GNC verification and validation). This SW version, that may still run in a general-purpose desktop PC or laptop, is used for profiling analysis. The code is compiled adding different debugging options that will add profiling instrumentalized code to get different execution and memory usage figures. On one hand, the code is used to evaluate the memory needed and the data quantity being interchanged (impacting memory access or input/output communication buses load). Secondly, this code is used to evaluate the performances onto the selected final flight HW avionics processing system. For example, if the execution time is violated, then a more powerful device is needed, or it should be considered to add a coprocessor to offload the processing system. Matlab or C implementation can be used for profiling. This prototype code for profiling might serve as the initial version of the final implementation, some parts would need adaptations, additions, modifications, recoding, or porting to a different language for HW development or DSP/GPU coding if a coprocessor is used.

Thanks to the profiling, a more robust decision on the avionics processing elements is taken. On the other hand, it may happen that the avionic components selection is specified by the client at requirements level, and, in this case, the task is to design and adapt the GNC solution and implementation to the requirements, considering the selected platform. In any of the two scenarios, the flight avionics are identified and a more refined design and test, in the form of feasibility analysis and schedulability analysis can be performed. In case of implementing the GNC (all or main part) in SW targeting a space processor, we can extrapolate performances metrics of the prototype onto the flight computer. From several processors and from what the client configures of system clocks and CPU load margins, performance values can be obtained in millions of instructions per second. Dhrystone million instructions per second (DMIPS) is the Dhrystone benchmarking of various functionalities executed on the processor to obtain times and give a

comparable value. Another benchmark that is beginning to be widely used is the CoreMark. The advantage of this benchmark is that it produces a single-number score allowing users to make quick comparisons between processors. Knowing the performance of the prototype and knowing the HW performances of the processor used during a test analysis, the developer can compare the benchmarking figures on two different processors with the execution time of the prototype code. If, for instance, a test of the prototype GNC SW on a Leon3 core with a performance of 106 DMIPs and it takes 2 s, it can be extrapolated that on a Leon4 core with a performance of 425 DMIPs and a requirement for 50% of CPU load margin (212 DMIPs equivalent), the time of your code would take about 1 s for that same function. These numbers are very theoretical and should be used as first rough indication especially in comparisons of processors with very different architectures (e.g., an Intel OctaCore and an Arduino processors). It should also be considered if the code is executed on one operating system or another, due to the overhead it could add. Nevertheless, it can be used as first feasibility analysis that can be used for a schedulability analysis by the SW engineers of the system. If available, an emulator of the target flight processor can be used to run the application SW instead of the actual processor or a representative processor with similar characteristics to derive further analysis. Another important aspect to consider during on-board avionics verification and testing is the selection of a proper margin philosophy. This is needed to ensure more resources and time than currently needed, to still have margin in case some SW parts are still on-going or new functionalities are included. The margin philosophy can be applied in SW as a CPU load percentage, as an execution time percentage penalty, as a memory utilization margin, or as a communication bus load margin. In HW solutions like FPGA-based, the margins are applied to the FPGA-utilized resources to implement a function and clock frequency margin.

A necessary analysis during the validation of on-board avionics is the schedulability analysis. The schedulability analysis consists of the definition of each piece of SW task periodicity, latency, and execution time, based on the real events timing in the environment and in accordance with all the tasks to be executed in the processor. For each task, a periodicity of execution time before changing to another task is defined. The functions of the tasks do not explicitly require executing in one periodicity execution window; therefore, the task functions can be executed in two time slots, providing half the full function time (as soon as does not create a misalignment problem or non-atomic execution problem).

The proof of the correct behavior and performances of the on-board implementation is provided, thanks to the Validation and Verification (V&V) campaign. In this context, we will use the term Validation to identify tests, analysis, inspection, or review of the implementation with respect to the requirements. The Verification, instead, is the process to verify and prove that the results of the SW implementation functions are correct with respect to the specifications and inputs of the system/subsystem/module.

The final on-board implementation may follow a dedicated hand-written coding or techniques for automatic code generation (autocoding) from an architectural model. For hand-written coding process, a technical specification should be adopted and followed. Moreover, it is possible and even convenient to use the architectural or mathematical model prototype as a reference. For autocoding approach, the original models should be properly configured or modified with certain restrictions and characteristics to allow the autocoding for real-time SW solution of critical nature as the GNC. In both hand-written or autocoding versions, the implementation shall consider the timing and synchronization mechanism. During this task, dedicated SW engineers should take the leading role for the coding of each SW unit, documentation, unitary verification, and build procedures to compile and link SW units. Once the code is implemented for the first time into its final version, a first GNC functional verification might be accomplished, thanks to a processor-in-the-loop (PIL) (see [Chapter 12](#) – GNC verification and validation) campaign. The PIL verification might only involve the application SW and a representative processor HW, normally not yet the final flight HW. Other HW elements such as sensors or actuators might still not be present in a PIL campaign, and values will be provided to the processor and the SW functions in the processor, thanks to an SW simulator or an HW emulator, in a representative equivalent way as if the actual sensors and actuators were connected. The interface with the SW simulator or HW emulator might not be the final actual interface with sensors and actuators, but the data volume and periodicity/latencies should be the same or representative of the final conditions. PIL campaign extends pure SIL functional verification including proper management functions, including timing and performance metrics verification, and functional nominal tests plus robustness cases introducing errors into the system to verify its response in those conditions, as per the specification defined. The application SW implementation and its PIL V&V may take into account the sensor data input volume, the number of sensors interfaced, the

frequency of each sensor data transferred, and its communication bandwidth. The output of the GNC application SW implemented shall consider the needed frequency and processing time latency for each epoch time step of the GNC function execution. The GNC itself might include different subfunctions that will be organized in the operating system in different tasks at different frequency. If navigation and control functions are scheduled at different rates, navigation shall consider output propagation to interface with the control and vice versa for every execution of the highest frequency task. If sensor data are not received at navigation and control frequency rate, again, proper propagation of navigation output shall be considered. A function to manage potential problems in the SW execution shall be designed and implemented considering different errors and associated risks during the code execution and considering HW malfunctioning of the processor, the OBSW, or the sensors/actuators suite. The OBSW might include this kind of functions, normally under the FDIR module (see [Chapter 11 – FDIR development approaches in space systems](#)). The GNC application SW shall either implement total FDIR for GNC or support to central SW FDIR.

The V&V activities to be performed should follow the agreement and description provided in a system or SW V&V plans documents. The Validation task should include a statement of compliance to the technical specification and baseline/derived system/subsystem requirements. Proper identification and traceability of V&V proof shall be documented. The verification activities shall include internal reviews on technical documents, reviews on design and code, inspection on received and developed items. The verification activities in the on-board implementation shall also include actual testing covering unitary and integration test cases, system test cases, validation and acceptance test cases, test coverage review, and test results review and analysis.

Test cases shall be determined to cover the compliance to requirements and specification. The test cases include input data, expected output data, internal modifications or events triggered, and test procedures. The test cases shall cover algorithmic functionality and performance metrics. For the developed SW or FPGA code, the test cases shall also verify the agreed code coverage percentage (depending on project or agreement with client it might be 90%, 99.7%, or 100%). For the code coverage, a profiled code is executed during the test campaigns to prove that every SW or FPGA code line is executed under the V&V requirements conditions.

The unit testing is the execution of test cases to validate isolated and individual pieces of SW code, functions, or subsystem. It may include white box and black box test, i.e., tests knowing the content of the functions to verify and tests modules where only input and outputs are known by the verification team. Different SW simulation or debugging tools shall be used for these purposes. As well as for the rest of the test campaign, the unitary and integration tests shall be repeatable and automated.

Integration tests will be executed after unitary test or even in parallel. Integration tests are normally designed in an incremental manner, having the purpose of gathering in subsequent tests the V&V of an incremental combination and interaction of different previously verified unitary functions, SW components, HW components, or both. To build this incremental process, stubs or wrappers can be used with the focus of providing all the interfacing connections to the modules under test but without adding functionalities that might not yet be verified at the same time to allow isolate the V&V incremental purposes.

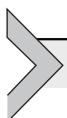
A special testing case campaign is the regression testing. It should be applied after a modification on an already validated SW, module, subsystem or system, repeating test cases or adding new ones to verify that after the changes the tests are still in line with the V&V requirements and that they do not create other undesired effects.

The final V&V of the whole on-board implementation is performed lately by system test campaign that include all the SW and HW functions of the subsystem as a black box test covering all the requirements and testing scenarios as defined and agreed in the plans and specification of subsystem. A subset of system test campaign might be defined as the acceptance tests, which are defined by customer for the approval and acceptance of the implementation delivery output product. Please refer to HW-in-the-loop HIL approach (see [Chapter 12 – GNC verification and validation](#)) for the specific accommodation of System tests using target HW processing devices, real interfaces, real sensors, and in a relevant environment, scenario, and conditions with respect to the mission.

References

- [1] <https://www.microsemi.com/>.
- [2] SpaceWire Proceedings, in: <http://2018.spacewire-conference.org/downloads/2018SpWProceedingsnew.pdf>, 2018.
- [3] Exomars 2022 Mission, <https://directory.eoportal.org/web/eoportal/satellite-missions/e/exomars-2022>.

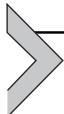
- [4] Exomars Control Center. <https://www.gmv.com/en/Company/Communication/News/2019/06/ExomarsCC.html>.
- [5] Experience Summary on Microsemi RTG4 Designs, SEFUW: SpacE FPGA Users Workshop, fourth ed., Johannes both, Edgar Kolbe.
- [6] H. Sierks, H.U. Keller, R. Jaumann, et al., The Dawn framing camera, Space Science Reviews 163 (2011) 263–327, <https://doi.org/10.1007/s11214-011-9745-4>.
- [7] J.P. Cobos Carrascosa, B. Aparicio del Moral, J.L. Ramos Mas, M. Balaguer, A.C. López Jiménez, J.C. del Toro Iniesta, The RTE inversion on FPGA aboard the solar orbiter PHI instrument, in: Proceedings Volume 9913, vol 991342, Software and Cyberinfrastructure for Astronomy IV, 2016, <https://doi.org/10.1117/12.2232332>.
- [8] P. Bajanaru, R. Domingo, D. Gonzalez-Arjona, F.A. Stancu, C. Onofrei, M. Marugan, R. Chamoso, C.G. Mihalache, Reconfigurable Co-processor for Spacecraft Autonomous Navigation”, European Workshop on On-Board Data Processing (OBDP2021), ESA-ESTEC, The Netherlands, 14–17 June 2021.
- [9] P. Bajanaru, D. Gonzalez-Arjona, F.A. Stancu, A. Alexe, D. Gogu, S. Sincan, O. Dubois-Matra, J. Alves, S. Vijendran, On-board complex image-processing based on FPGA acceleration for autonomous navigation in space, in: European Workshop on On-Board Data Processing (OBDP2019), ESA-ESTEC, The Netherlands, 25–27 February 2019.
- [10] Fall, Xcell Journal, Xilinx, FPGAs on Mars, David Ratter, Field Applications Engineer, Un Horizons Electronics, 2004.
- [11] Mars 2020 Perseverance Landing Press Kit. https://www.jpl.nasa.gov/news/press_kits/mars_2020/landing/mission/spacecraft/perseverance_rover/.
- [12] D. Gogu, F. Stancu, A. Pastor, D. Fortun, D. Gonzalez-Arjona, O. Müler, M. Barbelian, V. Pana, Boosting Autonomous Navigation Solution Based on Deep Learning Using New Rad-Tol Kintex Ultrascale FPGA”, European Workshop on On-Board Data Processing (OBDP2021), ESA-ESTEC, The Netherlands, 14–17 June 2021.
- [13] State-of-the-Art Small Spacecraft Technology Small Spacecraft Systems Virtual Institute Ames Research Center, Moffett Field, California, October 2020. NASA/TP—2020—5008734.



PART THREE

AI and modern applications

This page intentionally left blank



Applicative GNC cases and examples

Stefano Silvestrini¹, Andrea Colagrossi¹, Emanuele Paolini²,
Aureliano Rivolta², Andrea Capannolo¹, Vincenzo Pesce³,
Shyam Bhaskaran⁴, Francesco Sanfedino⁵, Daniel Alazard⁶

¹Politecnico di Milano, Milan, Italy

²D-Orbit, Fino Mornasco, Italy

³Airbus D&S Advanced Studies, Toulouse, France

⁴NASA Jet Propulsion Laboratory, Pasadena, CA, United States

⁵ISAE-SUPAERO, Toulouse, France

⁶ISAE-SUPAERO, Toulouse, France

The chapter presents a set of applicative guidance, navigation, and control (GNC) examples and use cases covering the topics of GNC that are of relevance for practical and modern applications. The content described here serves to decline the founding concepts covered in the previous parts of the book to different applicative cases, showing an end-to-end workflow to design the GNC system, or a part of it.

The proposed examples cover orbital and attitude control system (ACS) design, relative GNC applications, methods for on-board sensor processing, techniques for missions flying around irregular Solar System bodies, and planetary landing. Moreover, an example Attitude and Orbit Control System (AOCS) design process is used to introduce the chapter and to apply the GNC design methods, from requirements to preliminary design.

The covered topics span the entire applicative spectrum of modern spacecraft GNC, and, for this reason, the different sections of this chapter belong to different domains and have a broad variety of terminology and theoretical concepts. The reader is invited to read each section as an autonomous block, linking the presented concepts to the referenced parts of the book. Furthermore, the list of covered applications is obviously not exhaustive but contains those examples that are deemed to be more relevant for the modern challenges faced, on a day-by-day basis by the spacecraft GNC engineers.

The content of this chapter is structured as follows:

- *AOCS design.* This section summarizes the GNC design process, introduced in [Chapter 1](#) — Introduction and discussed along the whole

- book, with an applicative example on an AOCS. The system-level trade-offs, the AOCS modes, and the definition of the control types are discussed. The section is concluded with some comments on the selection of sensors and on the sizing of actuators for the example AOCS design.
- *Orbital control systems.* In this section, the most relevant orbital control methods are discussed, differentiating between impulsive and low-thrust control. The most common orbital maneuvers are presented, together with a brief section on the Lambert's problem for orbital targeting applications. Finally, the fundamental principles behind low-thrust trajectory design with optimal control methods and station-keeping for orbital maintenance are introduced.
 - *Attitude control systems.* This section presents some of the most useful methods to deal with spacecraft attitude control, such as detumbling, one-axis pointing, and three-axis pointing. Specific control techniques with reaction wheels and magnetorquers are presented, including a short section on reaction wheels desaturation. An applicative emergency solar panels pointing control is proposed, and a robust attitude control to deal with flexible appendages is preliminary designed at the end of the section.
 - *Relative GNC.* This section presents some techniques to deal with GNC systems for relative and proximity operations. Trajectory design, guidance, and control strategies are discussed. Finally, an example operative case of a rendezvous in cislunar space is presented.
 - *On-board sensor processing.* Some methods for on-board sensor processing are discussed in this section, with particular focus on failure detection, isolation, and recovery (FDIR) applications. Moreover, two applicative examples to be performed on-board for autonomous sensor calibration and Global Navigation Satellite System-Inertial Navigation System (GNSS-INS) orbit determination are presented.
 - *Irregular Solar System bodies fly around.* This section discusses the most relevant GNC aspects related with the fly around of irregular Solar System objects.
 - *GNC for planetary landing.* This section proposes an entire GNC system to deal with the problem of safely landing a spacecraft on the surface of a planet. The focus of the section is on the powered descent phase, and it does not account for the planetary atmosphere.



AOCS design

GNC is a wide terminology that has several major fields of application and can be used for any controlled dynamic system implying the functions of estimating the state, provide a reference, and control the estimated state to follow the reference one. On the other hand, the term AOCS is specific to spacecraft design: it is commonly used when the orbit guidance is not performed on-board, which is the case for standard low Earth orbit (LEO) and geostationary Earth orbit missions. When used in relation to spacecraft, the term GNC is typically used for the on-board segment when the satellite position is controlled in closed loop, for instance, in case of rendezvous and formation flying.

Regardless the specific terminology, the concept behind GNC and AOCS is analogous, and the processes to design a GNC system or an AOCS are closely related and follow the same steps. This section discusses the process to design and define the AOCS subsystem for a generic Earth-orbiting spacecraft. However, the same approach and concepts can be easily applied and extended to any GNC subsystem for modern spacecraft applications.

AOCS design process and subsystem architecture

The AOCS design process is iterative and involves the following steps, summarized in Fig. 14.1:

- Definition of high-level requirements with system engineering, starting from mission and payload requirements, and including all the high-level functionalities of the spacecraft that involve the AOCS subsystem, such as operative states, required accuracies, stability, etc. This includes also support to the definition of budgets for pointing accuracy, orbit control precision, and mass-volume allocation. The output of this phase is a set of requirements providing a clear idea of what the AOCS subsystem is called to do.

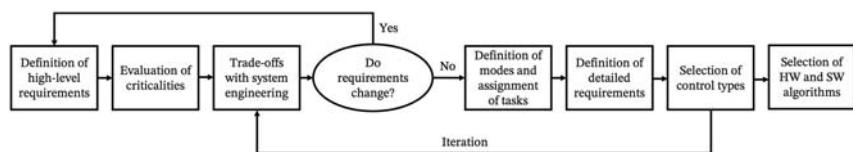


Figure 14.1 AOCS design process.

- Evaluation of criticalities possibly impacting high-level system design. Relevant disturbances, how required orbital and attitude states impact the subsystem, what is needed to achieve the required performances, interfaces, impact on mission operations, constraints imposed to the overall system and to the other subsystems, etc.
- Trade-offs with system engineering. This will be deepened in this section, and it could result in new possible requirements or requirements change for AOCS design.
- Definition of AOCS modes, subdivision of tasks among them, and outline of the AOCS state machine, which will be further discussed later in this section.
- Definition of detailed subsystem requirements for each AOCS mode.
- Selection of control types for each mode, in order to fulfill the relevant and applicable requirements.
- Iteration of the previous steps.
- Selection of hardware (HW) and software (SW) algorithms.

Once the high-level design is completed, then the detailed design can further proceed with the following steps:

- Design and development of the AOCS SW functions and assessment of the available performances in nominal and nonnominal cases, sensitivity, and robustness analysis.
- Verification, validation, and testing, both at HW and SW level.

The AOCS design shall consider the structure of the entire subsystem, which is typically formalized and detailed in an AOCS architecture. A typical AOCS architecture is shown in Fig. 14.2. The plant dynamics, in terms of instantaneous orbital and attitude states, velocity and angular rate, is sampled by sensors, whose data feed the navigation block that computes the estimated state of the satellite through data merging and filtering techniques, discussed in Chapter 9 — Navigation. The guidance is in charge of providing the desired state of the spacecraft, as outlined in Chapter 8 — Guidance, and the difference between the guided state and the navigated one feeds the controller, presented in Chapter 10 — Control, whose aim is to reduce this difference, so that the orbital and attitude states can reach or follow the desired targets. Telemetries and telecommands (TCs) from ground can eventually provide additional information in the loop, for example, in the case of guidance profiles loaded by ground, or when the configuration of the controller needs to be changed through updatable parameters and flags. Note that a proper GNC subsystem has its main difference with respect to an AOCS in this part, since the guidance profiles are

always computed on-board, and they are not uploaded with TC. The computed control output feeds the actuators, whose action has an impact on the spacecraft plant dynamics, together with the disturbing actions of internal and external factors characterizing the space environment, presented in [Chapter 3 — The Space Environment](#). Sensors and actuators data are an input also for the FDIR function that checks those data to mark each one as valid or not valid and autonomously decide which are the best navigation and control strategies based on data and component availability or unavailability. Higher level FDIR blocks can also check for major anomalies as, for example, a not correct Sun pointing or an actuator failure, leading to the possibility of an AOCS mode transition ([Fig. 14.2](#)).

Evaluation of criticalities

Right after the definition of the high-level requirements the AOCS shall fulfill, it is good practice to study how the overall system architecture impacts the AOCS design and vice versa, how the latter can impact or impose constraints on the system or other subsystems design. For example, if the payload has a high power demand, and thus large deployable solar arrays are foreseen by system high-level design, the AOCS designers should derive constraints for the structural design, in a way to limit the frequency of resonance of the panels to be 10 times larger than the desired control bandwidth. Evaluation of environmental disturbances is also important to understand if major design drivers can arise. For instance, drag can highly affect the duration of missions in LEOs. Therefore, outlining suitable low-drag attitude

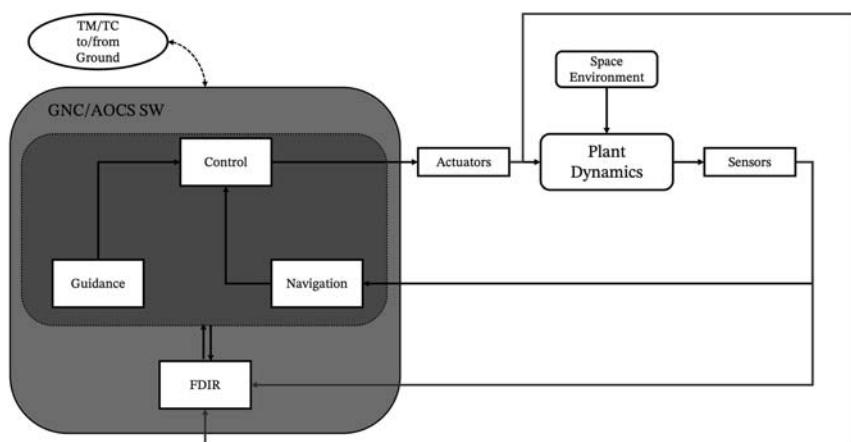


Figure 14.2 AOCS architecture.

states for the nonoperative phases of the mission can be vital to reduce the number of maneuvers to raise the altitude of the orbit in order to compensate for the atmospheric decay, which typically has a huge impact on the propellant to be carried on-board.

These are just two examples: evaluation of criticalities is very mission-specific, and it shall be evaluated case by case, under a system engineering point of view.

System-level trade-offs

When designing an AOCS system, it is of utmost importance to perform the necessary trade-offs with system engineering to decide high-level features and, through an iterative process, freeze the overall spacecraft configuration, satisfying the imposed high-level requirements and minimizing the impact of the criticalities on the mission. One of the main trade-offs is to decide the orbital and attitude states of the spacecraft during the mission: indeed, if on one side the dynamical states during payload operations are constrained by the payload itself, the same could be not true for other mission phases and alternative AOCS modes. Hence, in many cases, there is room to decide, for instance, if it is better to keep the satellite always in operative state, aligning the solar panels to the Sun, thanks to the action of the solar arrays drive mechanisms (SADMs), or to rotate the entire spacecraft to point the panels toward the Sun. The latter option could lead to smaller solar arrays, save the mass and the cost of the SADM, and bring higher flexibility, but with the drawback of more complex operations and higher spacecraft maneuverability.

These design choices really depend on a case-by-case basis, so that it is not possible to provide a single answer that is always valid. What a GNC designer shall understand is to perform the correct preliminary design analyses, taking into consideration all the possible factors, criticalities, and constraints. This process clearly has a broad range, well outside the GNC subsystem itself, in order to make the most correct design choices for the mission and avoid undesirable major changes in the architecture at a later stage of design. Another typical example of high-level trade-off is ground-based guidance versus an autonomous one; namely, the selection of an AOCS or a GNC subsystem. Similarly, a ground-based orbit control strategy versus an autonomous orbital station-keeping control, or even the choice of the control methods, or the algorithms for the different AOCS modes are other examples of high-level trade-offs.

A good approach to perform these design trade-offs is to summarize the pros and cons of each different option in a table and assign a vote for each one and for each of their relevant design feature. Table 14.1 shows an example of this design trade-off approach, where three alternative design options are compared: each of them is assigned a vote from 0 to 1 for each relevant feature. Note that each feature is differently weighted on the final sum, based on the real impact that it has on the spacecraft and on the mission. The option with the highest weighted mark is the best from an overall point of view. This means that it will not be at the same time the cheapest solution, the simplest at a technical level, the less impacting on the operations side, etc., but it is the one that from a global point of view brings the more advantages with, at the same time, the most acceptable drawbacks.

It shall be noted that this criteria matrix-based design is strongly affected by the weights and the policies to assign the votes that are inserted in the trade-off matrix, together with the features that are accounted in the trade-off process. Hence, the definition of the criteria matrix itself is a crucial part of the design, it cannot be generalized to different missions and it shall be agreed and discussed at system level.

Definition of AOCS modes

An important aspect of AOCS design is the division of tasks among different modes. GNC functions shall be tailored and tuned to achieve specific goals for the spacecraft, and this leads to the importance of having different navigation, guidance, and control functions depending on the AOCS mode. Thus, different specific AOCS modes can be implemented to facilitate the accomplishment of any task the spacecraft is called to carry out in each of

Table 14.1 Design criteria matrix.

	Weight	Option 1	Option 2	Option 3
Feature A (i.e., impact on operations)	20	0	1	0.4
Feature B (i.e., impact on interfaces)	30	0.6	1	0.8
Feature C (i.e., tech effort to implement it)	10	1	0.3	1
Feature D (i.e., cost)	40	0.7	0.5	1
Total		56	73	82

its operative phases. For example, a safe mode only uses the most reliable HW available on-board and needs very simple SW functions, with the acceptable drawback of a rough pointing, typically with the solar panels toward the Sun to recharge the batteries. On the other hand, an operative mode shall achieve the pointing performance required by the mission and by the payload. Therefore, it is important that the involved HW and GNC functions are appropriate to achieve that goal. For these reasons, the operative modes mostly drive the AOCS cost, while the safe modes impose reliability and efficiency requirements on the components and on the GNC SW. Examples of typical AOCS modes are reported in [Table 14.2](#), together with their main tasks and characteristics.

AOCS modes can differ from system modes, sometimes called system states: when the spacecraft is in operative state, AOCS could be in stand-by mode, or in operative, or in orbit control mode. Typically, when the spacecraft goes into safe state, it drives AOCS to go in safe mode, but the vice versa is not always recommended. In general, the transitions between the different modes are regulated according to a finite-state machine (FSM).

An FSM is a mathematical model of an abstract machine describing the behavior of an automated system, such as a complete AOCS/GNC subsystem with multiple modes, or an entire spacecraft system. An FSM can be in exactly one of a finite number of states at any given time, and it changes from one state to another in response to some inputs; the change from one state to another is called a transition. Transitions can be autonomous, automatic, or commanded from ground. Note that automatic typically refers to transitions that follow very strict boundaries defining the scope and the parameters of the transition. In this case, the automatic transitions happen according to a well-defined set of predefined events. Autonomous transitions are performed according to a decision of the system, which learns and adapts to dynamic environments, and evolves as the environment around it changes. These concepts are discussed with more details in [Chapter 1](#) — Introduction. An FSM is defined by a list of its states, its initial state, and the inputs that trigger each transition. [Fig. 14.3](#) shows a simplified example of an FSM of a satellite, including typical nonnominal FDIR transitions.

Definition of control types

Once the AOCS modes have been established and detailed, and the requirements for each of them have been finalized, control types can be individually selected for each mode. The aim of AOCS and GNC is to overcome both

Table 14.2 Typical AOCS modes.

AOCS mode	Tasks	Characteristics
Safe hold mode	<ul style="list-style-type: none"> • Detumble the satellite • Point the solar panels toward the Sun 	<ul style="list-style-type: none"> • Most reliable hardware shall be used • Power consumption shall be minimized • Rough pointing is acceptable • It shall be completely autonomous
Stand-by mode	<ul style="list-style-type: none"> • Point the solar panels toward the Sun • Point antennas toward the Earth (i.e., ground station) • Minimize drag during nonoperative phases • Point radiators to deep space for thermal control • Others depending on the mission 	<ul style="list-style-type: none"> • Nominal hardware should be involved (i.e., more complex than in safe mode) • Improved pointing performance with respect to safe mode, but it could be worse than in operative mode • It is typically autonomous • GNC functions should be different from the ones used in safe mode
Operative mode	<ul style="list-style-type: none"> • Accomplish the main goal and satisfy the primary requirements of the mission (e.g., image acquisition, antenna pointing, ...) 	<ul style="list-style-type: none"> • Most precise hardware is involved • Most precise pointing is achieved, using complex filtering or control techniques • It is typically ground-assisted
Orbit control mode	<ul style="list-style-type: none"> • Perform orbital maneuvers • Point and stabilize the thrusters in the desired direction 	<ul style="list-style-type: none"> • Propulsion is involved • Precise and stable pointing is achieved • Thrust parasitic torques shall be counteracted • It is typically ground-assisted
Transfer mode	<ul style="list-style-type: none"> • Control the attitude and the orbit during long transfers (e.g., along geostationary transfer orbits, or interplanetary orbits) 	<ul style="list-style-type: none"> • Propulsion is involved • Precise and stable pointing is achieved • Thrust parasitic torques shall be counteracted • It is typically autonomous
Mission mode	<ul style="list-style-type: none"> • Achieve particular goals and targets of the mission 	<ul style="list-style-type: none"> • Thermal and power requirements shall be fulfilled • Various

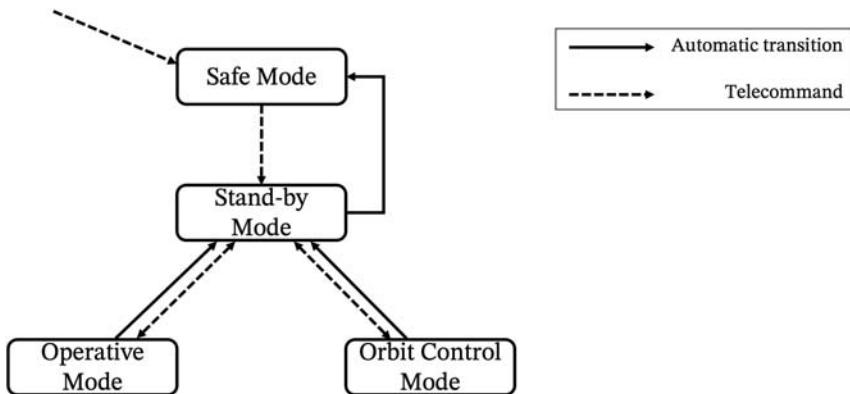


Figure 14.3 AOCS state machine and mode transitions.

internal and external disturbance torques and to appropriately control the spacecraft dynamics to achieve the mission goals. Control types can be mainly divided into passive and active.

Passive control techniques are those in which the spacecraft is inherently stable; therefore, no GNC control action is required. Examples of these kind are spin stabilization, where stability of the spinning axis in the inertial frame is granted, thanks to the conservation of angular momentum; gravity gradient stabilization, where elongated spacecrafts are stabilized along the local vertical direction, thanks to the different action of the gravity on the portion of satellite closer to Earth with respect to the farther ones; magnetic stabilization through the use of permanent magnets; frozen orbit stabilization, where the orbital state variations are minimized, thanks to a proper selection of the orbital parameters; Earth oblateness Sun aspect angle stabilization, where the orbital precession is controlled by selecting the orbital inclination in a way to minimize the apparent Sun motion with respect to the orbital plane (e.g., Sun-synchronous orbit). However, passive control techniques can achieve only rough accuracy, and they also impose strong constraints on the space of admissible dynamical states.

Thus, the most precise missions will certainly require for active control techniques, where the spacecraft orbital and attitude states are controlled by means of actuators that provide controlled forces and torques on the plant dynamics. Active orbital control can be used to impose boundary limits on the orbital parameters to perform station-keeping, or it can change the spacecraft orbit to achieve orbital transfers or to modify the spacecraft ground tracks. Analogously, it can be used to perform relative orbital control

during rendezvous or formation flying missions. Active attitude control is commonly used to control the directions of the spacecraft axes, and it can be momentum biased, controlling one or more axes, or zero momentum, controlling the three spacecraft's axes. The active attitude control definition depends on the resulting momentum created on the satellite. Momentum-biased techniques use momentum wheels to create a momentum vector that inherently stabilize an axis of the satellite, which will be the one with the sensors or the payload to operate, and it is often used for local vertical (i.e., nadir) pointing or other static pointing modes. Zero momentum has no restriction on the pointing that can be achieved; it can provide fast slews since no momentum vector shall be rotated, and it normally reaches the highest accuracy since it is able to quickly react to any disturbance. However, it needs attitude information on all the three axes, and it is usually the most expensive type of control. A zero-momentum configuration can be reached indifferently with thrusters, reaction wheels, magnetic torquers, or control moment gyros (CMGs). The specific actuator to be used shall be evaluated case by case, considering the necessary torque and momentum storage during the required slews, the lifetime (e.g., thrusters are limited by propellant, wheels, and CMG mainly by bearings duration), the costs, etc.

The most basic controller to achieve active control is of proportional type, as explained in [Chapter 10 – Control](#). In this case, the proportional gain K_p can be selected in order to limit the steady-state errors to the desired value:

$$K_p \cong D/\epsilon,$$

where D is the disturbance term and ϵ the desired steady-state error. K_p also defines the controller bandwidth that is given for the attitude control case by:

$$\omega_n \cong \sqrt{K_p/I}$$

given I the inertia of the spacecraft. The bandwidth defines the frequency at which the control action starts to be less effective (i.e., the control authority will be effective from 0 frequency up to the bandwidth), and it can be easily generalized for the orbital control case. A derivative action can be added to increase the control stability, keeping the same steady-state accuracy, by increasing the state damping that will slow down the response of the controlled system. Moreover, the steady-state error can be further decreased with the addition of an integral term in the controller, coming at the cost of reduced stability margins. [Table 14.3](#) summarizes how AOCS requirements

can impact on the architectural choices of the attitude control subsystem and on the attitude control types.

Sensors selection and actuators sizing

Sensor selection comes from reverse engineering of the pointing and control budgets of the spacecraft. Indeed, sensor errors in terms of bias, low-frequency noise, and temporal noise provide an important figure of evaluation for AOCS design. In fact, these errors contribute to navigation error that is one of the contributors to the Attitude Knowledge Error (AKE) in the pointing and control budget. Note that, for the attitude sensor, the errors are commonly summed up in the NEA, the Noise Equivalent Angle.

For a preliminary error assessment, the GNC designer can make a residual sum of squares of the sensor errors, and then it can estimate the navigation error as:

$$\text{Nav Err} = \frac{\text{RSS}}{\sqrt{n}}$$

where n is the number of sensors used in hot redundancy. Nav Err could be worse in the case of nonorthogonality between the sensors' lines of sight, but this effect is typically negligible up to 70 degrees of angular separation between lines of sight (e.g., degradation could be 1-2 arcsec when the angle between lines of sight is 70 degrees, but then raises exponentially for smaller angles). Then, the navigation error sums up with other contributors in the pointing budget (i.e., thermomechanical deformations, misalignments, ...) to compute the AKE. Therefore, sensor selection is typically performed backwards starting from the required AKE, splitting it among its contributors, including the navigation error. In this way, a value for the sensor errors can be found to be compared with suppliers' specifications. As it can be guessed, this is an iterative process since other contributions, even from other subsystems, are involved.

Coarse sensors are widely used for AOCS rough control modes or for temporary backup solutions. In modern space missions, these sensors may be used as state acquisition sensors even in operative modes. Coarse Sun sensors, magnetometers, microelectromechanical system inertial sensors, and commercial GNSS receivers are examples of this category. When dealing with these sensors, it is important to accurately address noise as accurately as possible, in order to achieve the desired performance. Moreover, since they are frequently used in small satellite or low-cost missions, also mass-

Table 14.3 AOCS requirements and architectural choices for attitude determination and control.

Requirements	Effects on attitude determination	Effects on attitude control
Pointing error > 5 degrees	If gravity gradient (GG) stabilization is used, there is no need for attitude determination. Otherwise, Sun sensors and magnetometers are adequate.	Depending on the mission pointing modes, passive control could be feasible, even with GG stabilization, leading to major cost savings.
Pointing error from 1 to 5 degrees	Sun sensors, magnetometers, and horizon sensors are adequate.	GG stabilization would be too rough. Spin stabilization is feasible if the mission needs an inertially fixed attitude. Active control could be needed; in the case magnetic torquers are viable, the wheels are not required.
Pointing error from 0.1 to 1 degrees	Accurate attitude determination needed: coarse sensors are not viable. There is the need for star trackers, or horizon sensors, and gyroscopes depending on the mission.	Passive control is not possible. Both momentum bias and zero momentum are viable. Typical actuators are reaction wheels, with magnetic torquers or thrusters for momentum unloading and coarse control.
Pointing error <0.1 degrees	Star trackers and accurate gyroscopes are necessary.	Three-axis zero-momentum control is needed. Control of minor disturbances, such as flexible modes and sloshing is very important. Complex GNC algorithms are necessary.

(Continued)

Table 14.3 AOCS requirements and architectural choices for attitude determination and control.—cont'd

Requirements	Effects on attitude determination	Effects on attitude control
Angular Rates < 1 degrees/s	None	Reaction wheels are usually sufficient, depending on spacecraft inertia.
Rates >1 degree/s	Star trackers may be not adequate.	CMG or thrusters could be necessary for slews.

volume considerations are important, as well as radiation tolerance, especially if commercial components designed for ground applications are under evaluation.

Magnetic torquers are typically used for desaturation of momentum exchange actuators, and for magnetic detumbling of the satellite after release from the launcher. In such a case, the sizing is performed against the total angular momentum to be dissipated by the torquers in a prescribed amount of time, which is safely allowed by the spacecraft's batteries before entering a critical condition. Magnetic torquers are typically used in a duty cycle with magnetometers: when reaching the maximum duty cycle, torquers are switched off to leave enough time for dipole discharging and subsequent magnetometers data acquisition. This is to avoid magnetic interference, as discussed in [Chapter 7](#) — Actuators. [Table 14.4](#) reports a one-axis example

Table 14.4 Example magnetorquer sizing.

Parameter	Value	Unit measure
Orbital altitude	500	km
Mean magnetic field, B	30	μT
Spacecraft Inertia, I	300	$\text{kg}\cdot\text{m}^2$
Time allowed for detumbling, T_{all}	4	Orbits
Duty cycle with magnetic torquer switched off, DC	25	% of control cycle dedicated to discharge magnetic torque dipole and read magnetometer data
Design margin, M	20	%
Angular rate at release, ω	2.5	$^\circ/\text{s}$ at 3σ from launcher user manual
Momentum to be damped, H	13.08	Nms ($H=I\omega$)
Orbital period, T	5674	s
Time for damping, t_d	12,483	s ($t_d = T_{all} \cdot T \cdot [(100 - DC - M)/100]$)
Needed torque, C	0.001048	Nm ($C = H/t_d$)
Magnetic dipole, m	35	Am^2 ($m = C/B$)

of magnetic torquer sizing against detumbling of initial angular rates of an LEO spacecraft. Note that design margin, M , is meant here to account for variability of preliminary data such as inertia of the spacecraft, but also for the fact that the control torques are constrained to lie on the plane perpendicular to the local magnetic field direction.

Reaction wheels sizing is typically performed against agility requirements, which are associated to the maximum slew rates that can be achieved by the AOCS. A simple way is to consider bang–bang profiles for the attitude maneuvers, where 60%–70% of the maximum torque from reaction wheels is used, leaving the rest for disturbance compensation. Assuming θ as the slew angle, $\dot{\omega}$ as the desired angular acceleration, and t_{TOT} as the total maneuvering time, a maximum torque profile leads to:

$$\frac{\theta}{2} = \frac{1}{2}\dot{\omega}\left(\frac{t_{TOT}}{2}\right)^2 \rightarrow t_{TOT} = 2\sqrt{\frac{\theta}{\dot{\omega}}} \quad \text{or} \quad \dot{\omega} = \frac{4\theta}{t_{TOT}^2},$$

where half of the slew angle is covered in half of the prescribed time. For example, consider the following typical requirement: “The spacecraft shall be able to perform 90 degrees slews in less than 3 minutes.” This leads to a required angular acceleration of $0.011^\circ/\text{s}^2$ or $1.9e-4 \text{ rad/s}^2$. Assuming to leave 30% of the available torque for disturbance compensation, and assuming an inertia of 300 kgm^2 , it is obtained that a control torque of 0.058 Nm is required to comply with the requirement. Note that gyroscopic effects have been neglected, as it is reasonable to assume as far as the angular rates during the slews are not substantially high.

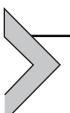
Thrusters can be used for orbit control or attitude control, and sometimes even for both. When thrusters are used for orbit control, then a trade-off between the ΔV to be delivered in a given amount of time (i.e., the need for larger thrusters to minimize time) versus the accuracy on the final orbit to reach (i.e., the need for smaller thrusters to maximize accuracy) shall be done. Typically, accuracy is more important than allocated time for orbital maneuvers; so, there is a wide tendency on using thrusts below 5 N, that is helpful also on the attitude control side, since parasitic torques during thrusters firing are typically relevant, as explained in [Chapter 3 – The Space Environment](#). When thrusters are used for attitude control, then a sizing for a detumbling or a fast slew is typically performed. Assuming an inertia of 300 kgm^2 , and an angular rate at release of $2.5^\circ/\text{s}$ on the same axis, then a momentum of 13.08 Nms shall be dissipated through the action of thrusters (cfr. [Table 14.4](#)). If a requirement asks for a maximum detumbling time of

two minutes, this turns in a request for a constant torque of 0.109 Nm. The needed thrust, F , can be computed recalling that, for two symmetric thrusters creating nominally a pure torque, t , around the axis:

$$t = 2Fd \rightarrow F = \frac{t}{2d}$$

where d is the arm between the application point of the thrust and center of mass of the satellite. So, to conclude the example, assuming an arm of 30 cm, two symmetric thrusters able to provide at least 0.182 N of thrust each are required.

CMGs are momentum exchange devices in which the spinning rotor is placed on a gimbal that can be rotated in the direction orthogonal to the spin. In this way, the momentum vector is changed in direction, resulting in a gyroscopic torque. Depending on the CMG model, the spin speed of the rotor can be either fixed or variable, and it can be rotated around one or two axes. Sizing of CMGs is a bit more elaborated to be treated here; however, it is widely discussed in Ref. [1].



Orbital control system

Orbital control is a broad term referring to control theory applied on the orbital dynamics. The goal of the orbital control system is to manage the orbital state to control the position and velocity of the spacecraft. As it was discussed in Chapter 4 – Orbital Dynamics, a body in space is always moving under the action of gravitational forces, which make practically impossible to have a static position control. Indeed, the spacecraft position and velocity are continuously changing in time as the spacecraft moves along its orbit. Thus, position control in space has always to be considered under a dynamical perspective, defining the control variables on the six orbital states of the spacecraft.

It shall be noted that often orbital control is not performed in closed loop. Many orbital maneuvers can be performed in open loop, and they are usually computed on-ground. Thus, many classical spacecrafts are just equipped with an AOCS subsystem, and the orbital control errors are progressively fixed by means of progressive correction maneuvers (e.g., trajectory correction maneuvers during interplanetary transfers). However, it shall be noted that the control budget allocated for correction maneuvers is remarkably smaller than the one for primary orbital insertion or orbit change maneuvers. Hence, the open-loop control shall not be confused with a

rough trial and error approach, since the typical spacecraft constraints leaves no room for orbital control errors. Even if the open-loop orbital control with the main GNC functions demanded to the ground segment (e.g., ground-based orbit determination, guidance and trajectory optimization, maneuver computation) is common for the largest orbital control operations, closed-loop system and on-board orbital GNC are not extremely uncommon. Moreover, they are becoming more popular for modern spacecraft, which are looking for increased autonomy and lower operating burden, and they are necessary whenever the real-time operations make the delay imposed by ground support not acceptable.

Despite the differences in the aforementioned approaches, the fundamental rules behind orbital control remain the same, and they can be applied to any of the previous cases. For this reason, this section preliminarily presents some of the most common and useful applicative examples about orbital control, regardless from the closed-loop or open-loop implementation, and from the location where these techniques are computed. Given the broad theory behind orbital control, the reader is strongly encouraged to deepen these concepts in the suggested literature references [51–54].

Impulsive and low-thrust maneuvers

Orbital control methods are strongly influenced by the performance of the thrusters used to actuate orbital control commands. In particular, the available thrust level divides between impulsive and low-thrust maneuvers.

In the first case, the thrusting force is so high that the acceleration applied to the spacecraft determines a velocity variation that is approximately instantaneous. That is, at a particular point in the orbit, the spacecraft changes the velocity from \mathbf{v}_1 to \mathbf{v}_2 . Reminding that an orbit is fully defined by the position and the velocity vectors, an impulsive maneuver makes the spacecraft dynamics to evolve from the current orbit to another one sharing a common point with the first. The change in velocity is given by:

$$\Delta\mathbf{v} = \mathbf{v}_2 - \mathbf{v}_1.$$

An important parameter of the maneuver is the magnitude of the velocity change:

$$\Delta v = \|\Delta\mathbf{v}\|,$$

which is a measure of the fuel consumption, and it is referred to as “delta-v.”

Minimum fuel maneuvers require minimum $\Delta\nu$, as discussed in [Chapter 7](#) — Actuators. Even if the fuel consumption is very important, the time of flight (TOF) (i.e., time required to complete a maneuver) shall not be overlooked. Obviously, impulsive maneuvers have $TOF = 0$, but complete orbital maneuvers may require more than one impulse. In this case, the overall TOF is not zero, and it accounts for the coasting phases to reach the location where the maneuvers shall be executed. It is not uncommon that a minimum $\Delta\nu$ transfer has a nonoptimal TOF, and a compromise between these two parameters is frequently required in orbital control applications.

This trade-off is not only valid for impulsive maneuvers but also it can be generalized to any kind of orbital control system. Indeed, the same concepts apply to low-thrust maneuvers, where the maneuver is not impulsive anymore (i.e., $TOF \neq 0$), and the $\Delta\nu$ is applied over a finite orbital arc, since the propulsive subsystem has a limited thrust level that produces a small control acceleration on the spacecraft dynamics. Hence, the velocity variation is continuous over the thrusting phase, and the orbital change is not related with a single location, but it distributed over the entire maneuvering time. Note that, strictly speaking, the low-thrust maneuver is only associated to the time when the thrusters are active, and the spacecraft is continuously changing its orbital state. However, the overall low-thrust orbital control makes use of both thrusting phases and coasting phases. The two are alternated, and the latter is typically exploited to optimize the effectiveness of the former. Indeed, the optimization of low-thrust orbital control is dedicated to select the best switching events to change between the two alternative phases.

Low-thrust control force is included in the spacecraft orbital dynamics as a perturbation force, since it can be considered smaller than the main gravitational attraction. Hence, the methods to deal with low thrust orbital control are based on the theory of orbital perturbations, which has been presented in [Chapter 4](#) — Orbital Dynamics. A common low-thrust design tool is based on the Gaussian variation of parameters equations. In particular, the equations for the semimajor axis $\frac{da}{dt}$, the eccentricity $\frac{de}{dt}$, and the inclination $\frac{di}{dt}$ are useful to understand the main impact of the orbital control force \mathbf{f} in terms of its components in $(\hat{\mathbf{r}}, \hat{\boldsymbol{\theta}}, \hat{\mathbf{h}})$: f_r, f_θ, f_h .

Note that no maneuver is really impulsive in the practice, but the method for impulsive maneuvers is valid whenever the thrusting time is much smaller than the orbital period (i.e., $t_{maneuver} \ll T$). If the thrusting

time is finite, the difference with respect to the ideal impulsive case can be accounted by considering the gravity losses. However, the impulsive maneuvers approximation gives acceptable results for many design and analysis purposes. On the contrary, low-thrust maneuver approach shall be only used when the thrusting time is long, and the control force is continuously applied over a finite arc of the orbit.

Orbital maneuvers

With the term orbital maneuvers, we classically refer to impulsive orbital maneuvers. At first, we will list and discuss some of the most common typologies of elementary maneuvers to have an insight on the fundamental concepts behind this topic. Then, the most general type of orbital maneuvers is presented, discussing the required solution of the Lambert's problem.

Coplanar maneuvers

Coplanar maneuvers affect the shape of the orbit in the orbital plane, and thus they are used to modify and control the semimajor axis, a , the eccentricity, e , and the argument of periaxis, ω .

The simplest and the most common coplanar maneuvers only consider tangential velocity changes. Namely, these maneuvers only change the velocity magnitude but not its direction. Thus, in general, the same position state will correspond in the new orbit to a different true anomaly, associated to another periaxis (i.e., argument of periaxis is changed). This is not happening if the tangential maneuver is applied in a point where the velocity is perpendicular to the orbital radius, and, recalling from the orbital dynamics equations, this occurs in circular orbits, or at periaxis and apoapsis of a generic orbit. Indeed, any tangential velocity change on a circular orbit results in that point becoming either periaxis or apoapsis of the new orbit. Accordingly, for an elliptical orbit, any tangential maneuver at periaxis results in a change of the height of apoapsis, while at apoapsis, it results in a change in the altitude of periaxis. Moreover, if we want to transform an elliptical orbit into a circular one using a tangential transfer, we can only do so at periaxis or apoapsis. Summarizing, a single tangential maneuver can only change the shape and dimension of the orbit (i.e., a and e) if performed in an apsidal point, while it can change shape, dimension, and orientation if executed in any other point.

Assuming the tangential maneuver is performed when the radius vector is, \mathbf{r} , the cost to perform the maneuver can be computed as:

$$\Delta v = \sqrt{\mu \left(\frac{2}{r} - \frac{1}{a_2} \right)} - \sqrt{\mu \left(\frac{2}{r} - \frac{1}{a_1} \right)},$$

where a_1 and a_2 are, respectively, the semimajor axes of the first and the second orbit. Note that if $\Delta v < 0$, the two terms in the previous equation shall be switched to get $\Delta v > 0$. In fact, a negative Δv means that the orbital velocity is decreased, but the maneuver is always associated to an active impulsive thrust. Hence, there is always a fuel consumption, and the only difference is the opposite thrusting direction. For this reason, the Δv is defined as a positive quantity.

If the original and the desired orbit have no point in common, a single maneuver would not suffice to reach the final orbital state. Indeed, in this case, an intermediate transfer orbit is needed to connect the two orbits. As a consequence, the spacecraft shall be controlled to move from the first to the transfer orbit, then a coasting phase is included until the spacecraft reaches the following maneuver point to get into the final orbit or into a further transfer orbit. The number of maneuvers to reach a desired orbital state is not constrained, and it is typically an output of the spacecraft trajectory design and optimization activities. The most common multiple maneuvers strategies are:

- Bitangent elliptic maneuver, with two maneuvering points.
- Bielliptic maneuver, with three maneuvering points.

Note that these maneuvering strategies assume the orbits to have the same line of apsides. The first one includes a special case, which is the Hohmann transfer. The Hohmann transfer is a bitangent maneuver between two coplanar circular orbits, and it can be proven that it is the minimum Δv double-impulse maneuver between coplanar circular orbits [52].

The Hohmann transfer consists of two tangential maneuvers: a circular to elliptical transfer followed by an elliptical to circular transfer. That is, a transfer from the first orbit to the final orbit is obtained through an elliptical transfer orbit with semimajor axis defined as:

$$a_t = \frac{r_1 + r_2}{2},$$

where r_1 is the radius of the departure circular orbit, and r_2 is the radius of the final one. The total velocity change for the Hohmann transfer is:

$$\begin{aligned}\Delta v &= \sqrt{\frac{\mu}{r_2}} - \sqrt{\mu \left(\frac{2}{r_2} - \frac{1}{a_t} \right)} + \sqrt{\mu \left(\frac{2}{r_1} - \frac{1}{a_t} \right)} - \sqrt{\frac{\mu}{r_1}} \\ &= \sqrt{\frac{\mu}{r_2}} \left[1 - \sqrt{\frac{2r_1}{r_1 + r_2}} \right] + \sqrt{\frac{\mu}{r_1}} \left[\sqrt{\frac{2r_2}{r_1 + r_2}} - 1 \right].\end{aligned}$$

The TOF for the Hohmann maneuver is half the period of the transfer ellipse.

Other coplanar maneuvers are possible, which are not limited to be tangent, but they can include variations in the velocity direction. In fact, by applying a thrust nonparallel to the velocity vector, it is possible to change the orientation of the orbital velocity. This opens to secant coplanar maneuvers, where the first and the final orbits have a cross intersection, and to maneuvers capable to change only the orientation of the orbit. Further details can be found in Refs. [52–54].

Plane change maneuvers

Other than the orbital geometry, and the orientation of the orbit in its plane, the orbital control may be intended to change the orientation of the orbital plane in the three-dimensional space. It is possible to control the orbital plane and the orbital geometry together, or the orbital plane alone. In this last case, the orbital control shall simply rotate the velocity vector about the position vector, in a way that the energy and the angular momentum of the orbit are unchanged.

If position and velocity vectors are perpendicular to each other, the velocity variation for a pure plane change is given by:

$$\Delta v = 2v \sin \frac{\theta}{2},$$

where θ is the rotation angle of the orbital velocity. Note that the Δv can be extremely high, and the plane change is typically one of the most fuel consuming orbital control maneuver to be performed by the GNC system. Therefore, if possible, the plane change should be performed as far as possible from the central attractor, where the orbital velocity reaches the minimum values.

In general, a plane change will alter both inclination and right ascension of the ascending node. If a pure inclination change is needed, the plane

change shall occur where the orbit crosses the equatorial plane. In generic case, the complete understanding of the maneuver is based on the spherical trigonometry to evaluate the velocity rotation angle and the maneuver location to achieve the desired final orbital plane. Moreover, it shall be noted that if position and velocity vector are not, respectively, perpendicular, the pure plane change $\Delta\nu$ is only influenced by the transversal velocity component.

It is worth noting a very relevant application of the plane change maneuver, which is the acquisition of the equatorial orbital plane. In fact, when a spacecraft is injected into orbit using a launcher, the point of orbital insertion is roughly above the launch site. Given the fact that the orbital plane must contain the center of the Earth and the insertion point, the minimum orbital inclination is approximately equal to the latitude of the launch site. Therefore, it is very difficult to launch a spacecraft directly into an equatorial orbit, unless the launch site is very close to the equator.

As said, if the orbit control has to change the size, shape, and plane of the orbit, a combination of the previous maneuvers is required. However, it shall be analyzed on a case-by-case basis, and it cannot be generalized. In general, such complex orbital maneuvers undergo a trajectory design and optimization process. The methods and the fundamental concepts behind orbital control optimization will be discussed in the following section about low-thrust trajectory design.

Lambert's problem

The most general type of orbital maneuvers connects a departure point to a final point. If the two points are not coincident, a transfer orbit shall be found. The orbital control system shall be capable to inject the spacecraft from the orbit containing the first point to the transfer one and then to acquire the final point when the transfer orbit reaches it. In general, the transfer orbit is not known and the GNC subsystem (i.e., the guidance functions) shall be capable to compute it.

The problem of finding the transfer orbit given two position vectors and imposing the TOF to travel between them is known as Lambert's problem. This problem basically consists of finding the orbit required to achieve a given transit time between two position vectors. It shall be noted that the Lambert's problem is only intended to find the transfer orbit satisfying the imposed position and time requirements, and it does not compute the necessary orbital control maneuvers. It is a crucial tool for GNC applications dedicated to orbital control, far-range rendezvous, and position targeting, but it

is also a fundamental element to solve preliminary orbit determination problems (i.e., given two observations with an interval of time between them, determine the compatible orbit).

We know that an orbit is fully defined once the position and velocity vectors are both specified. Thus, if we know the initial and the final position vectors (i.e., \mathbf{r}_1 and \mathbf{r}_2), we can find either the initial or the final velocity (i.e., \mathbf{v}_1 or \mathbf{v}_2), although \mathbf{v}_1 is more interesting for an orbital control system. In fact, this is the required spacecraft velocity at position \mathbf{r}_1 and time t_1 , in order to get at the desired position \mathbf{r}_2 at time $t_2 = t_1 + \text{TOF}$. According to the theorem of Lambert, the TOF from \mathbf{r}_1 to \mathbf{r}_2 is independent of the orbit's eccentricity and depends only on the sum, $r_1 + r_2$, of the magnitudes of the position vectors, the semimajor axis, a , and the length, c , of the chord joining \mathbf{r}_1 and \mathbf{r}_2 . It is noteworthy that the orbital period and the specific mechanical energy are also independent of the eccentricity. Moreover, two alternative solutions are available: one is prograde (i.e., $0^\circ < i < 90^\circ$) and the other is retrograde (i.e., $90^\circ < i < 180^\circ$). Generally, the prograde one is selected because most of spacecraft orbits are prograde and the maneuvering cost associated to this case is usually lower; although, the length of the transfer arc can be a further driver to select the most suitable alternative.

Different solutions to Lambert's problem are available and can be found in literature, and the reader is invited to read them in Refs. [51,54].

Low-thrust trajectory design

Low-thrust trajectory design wants to find a controlled trajectory that allows to reach a final point, from a departure orbital state, in a given interval of time. The difference with respect to the Lambert's problem is that the solution includes a control force and outlines the maneuvers to stay on the desired low-thrust trajectory. The control force is limited by the maximum thrust level, and the trajectory can be subject to additional constraints. Thus, the low-thrust maneuvers are commonly evaluated by solving a constrained optimization problem. Indeed, the low-thrust orbital control is a typical application of the optimal control theory and, to have a complete understanding of the matter, the reader is invited to familiarize with Hamiltonian formulations, Lagrange multipliers, and optimal control theory [55]. Note that the methods introduced here for low-thrust trajectory design and optimization can also be generalized and applied to impulsive maneuvers and classical orbital control systems.

The fundamental idea behind optimal control theory, applied to the orbital control problem, is to form the Hamiltonian of the system and evaluate the partial derivatives to compute the optimal solution for a desired set of cost functions. The Hamiltonian combines the state equations to the cost functions. Given that a typical orbital control problem has more cost functions, the Hamiltonian merges all the cost functions and equations of motion into a single, one-dimensional solution space. The state equations determine the dynamics and the boundaries of the system. These can be natural (e.g., gravity) or physical constraints (e.g., maximum thruster force). However, further constraints and restrictions can be included into the problem (e.g., keep-out zones). The cost functions, or performance indices, assess the optimality of the problem weighting certain conditions, which can be terminal (e.g., miss distance from a location) or cumulative (e.g., fuel usage, TOF). The weights and the cost function themselves may be a function of time or states. The overall optimization process is mathematically intensive and sometimes requires “human” reasoning to get to the proper solution. Thus, it is not easy to compute such a solution on-board, and these kind of orbital control systems are commonly ground-assisted.

The velocity change is accumulated In time during the thrusting phases. Δv_{LT} is the total velocity variation imparted by the thrust force during the elapsed control time, t_f :

$$\Delta v_{LT} = \int_0^{t_f} \frac{a_{LT}}{1 + \dot{m}_s t} dt.$$

where a_{LT} is the spacecraft acceleration, as imposed by the control thrust. The specific propellant mass flow rate, \dot{m}_s , is the actual mass flow rate divided by the initial mass of the vehicle; this value is negative for an active propellant consumption.

The low-thrust design solution defines the time history of the control force (i.e., the control acceleration) in a way that the spacecraft can reach the desired final state, satisfying the imposed constraints and minimizing the set of cost function, through the Hamiltonian of the system. The overall solution can include both thrusted arcs and coasting phases, where the dynamics is naturally left to evolve under the influence of the environmental forces. The switching between the two alternative phases can be used to better shape the optimal solution of the control problem, as further discussed in the suggested references [55].

The methods typically used to solve and optimize a constrained low-thrust orbital control problem are:

- Direct methods, which use a discretization of the trajectory in multiple path points and approximate the control with simple functions (e.g., constant, linear, polynomial, etc.) in each discretized arc. These methods perform a direct transcription of the optimal control problem and approximate the infinite-dimensional problem by a finite-dimensional one to solve it with nonlinear programming (NLP) algorithms. They exploit well-established and robust numerical methods, and they easily converge to a solution. However, they involve many parameters, and they are computationally intensive.
- Indirect methods, which are based on variational calculus and reduce the optimal control problem to a boundary value problem (BVP). These methods have a solid theoretical foundation, determining a limited number of parameters and a fast computation of the solution. However, they are poorly robust, and they are difficult to converge. In the presence of constraints, the optimal control solution is usually found applying the Pontryagin's maximum principle.
- Heuristic methods, which are based on laws inspired by nature and experience. These methods determine an optimal solution by iteratively trying to improve a candidate solution with respect to a given measure of quality, defined by a cost function. They make few or no assumptions about the problem, and they can search large spaces of candidate solutions, thanks to the modern computational capabilities. However, they do not guarantee either feasibility or optimality, and they commonly fail in understanding how close to optimality a particular feasible solution is. Different typologies of heuristic optimization algorithm exist, but the evolutionary and the particle swarm ones are the most suitable for orbital control.

It shall be reminded that low-thrust orbital control optimization is difficult to be computed on-board, especially if indirect and heuristic methods are used. In fact, they commonly require the supervision of an experienced GNC engineer. Direct methods are the most suitable for on-board applications, which are possible, thanks to the performance of modern on-board computers. Notwithstanding, the ground-based trajectory optimization is still the standard approach for orbital control systems. This is generally true for both low-thrust and impulsive-maneuver trajectories. Further details on the optimal control problem are discussed in [Chapter 8 – Guidance](#).

[Fig. 14.4](#) shows an example orbit-raising maneuver performed with a low-thrust orbital control system. The solution optimizes the fuel consumption, and it is found with an indirect method and a bang–bang control logic. Namely, the thruster is fully active along the thrusting arcs and switched off during the coasting phases. In this example, the thrusting direction is assumed to be always parallel to the spacecraft velocity vector. However, the optimization problem can also evaluate the best thrusting direction to further optimize the solution.

Station-keeping

In the previous section, the orbital control system was used and applied to control the spacecraft dynamics along orbital transfers and targeting applications. However, the on-board orbital control systems are also used to maintain the spacecraft on the nominal operational orbit, counteracting the effects of environmental perturbations. This task is normally denoted as station-keeping, and it is an orbital control application that can be fully

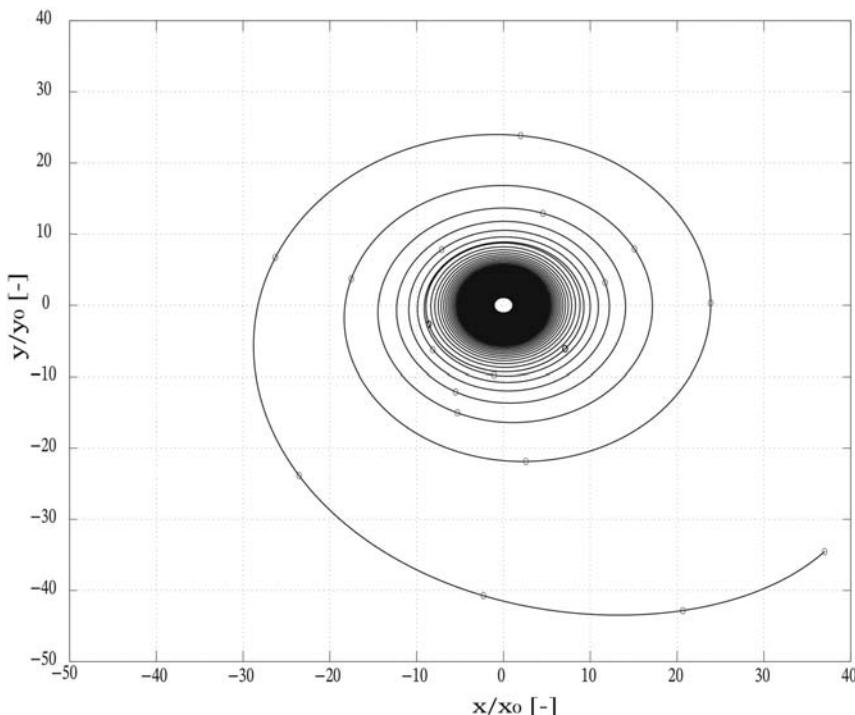


Figure 14.4 Orbit-raising low-thrust trajectory solution.

implemented on-board with a certain level of autonomy. In fact, differently for other orbital control problems, it is easier to define a feedback control loop on the errors with respect to the target Keplerian parameters and let the spacecraft compute the necessary maneuvers to maintain the desired orbital state. However, as always, there exists the possibility to have a ground-based GNC implementation, and the control commands are simply uploaded to the spacecraft.

Station-keeping is commonly not continuously executed, but it is periodically activated when the orbital state errors go out of the predefined limits, commonly defined as station-keeping box. Thus, the station-keeping control is applied with a dead band around the reference state. The station-keeping depends on the operational orbit, and it is related with the mission requirements since only few orbital elements may be of interest for the specific application. With reference to Earth-based spacecraft, the most common station-keeping orbital controls are:

- Drag compensation control is used in LEOs, and it is particularly relevant for space stations and below 400 km of altitude. In this case, the station-keeping is dedicated to control the orbital altitude, especially the pericenter's one, to avoid a shortening of the orbital period and a premature orbital decay with a catastrophic impact with the atmosphere. The Δv_{SK} for altitude maintenance is strongly dependent from the orbital radius and from the mass and the drag coefficient of the spacecraft (i.e., ballistic coefficient).
- Ground track maintenance is used to control the spacecraft passages over certain regions of the Earth, in a way to maintain a fixed and repeatable ground track, and the orbital period synchronous with the Earth's rotation. This station-keeping shall counteract also the faintest atmospheric drag and the luni-solar perturbation with effect on the orbital inclination. The latter is particularly important for Sun-synchronous orbits, which may require a Δv_{SK} in the order of 1–2 m/s per year to maintain a constant inclination.
- Geostationary North–South control is used to maintain the orbital plane inclination of geostationary orbits, which is perturbed by luni-solar attraction. The Δv_{SK} needed to compensate for this perturbation maintaining the orbital inclination on the equatorial plane is about 50 m/s per year.
- Geostationary East–West control is applied to keep the orbital period synchronous with the Earth' rotation and to keep a small orbital eccentricity. The former effect is due to the perturbations of the nonspherical

Earth, the latter is primarily due to the solar radiation pressure. The East–West station-keeping is less fuel consuming than the North–South one. In this case, $\Delta\nu_{SK}$ is in the order of 5 m/s per year.

- Lagrangian point station-keeping is applied in most of three-body problem orbits, since their stability is usually weak, especially when the orbit is about the collinear libration points (i.e., L1, L2, and L3). In this case, the orbital control is dedicated to cancel the unstable drifting component of the periodic dynamics (i.e., unstable manifold) to maintain the spacecraft on the prescribed trajectory. The $\Delta\nu_{SK}$ is typically very low, with an average value around 1 m/s per year.
- Relative station-keeping is applied for constellation and formation flying maintenance. This is a very specific orbital control dedicated to maintain the relative positions of the different elements in a constellation or a formation. However, the latter is commonly dealt with relative dynamics control, and it is managed by a dedicated GNC system. On the contrary, constellation maintenance is performed with a station-keeping action on each spacecraft element. The specifications and the cost (i.e., $\Delta\nu_{SK}$) of this station-keeping are dependent from the application, and it is hard to generalize their discussion. Thus, constellation maintenance station-keeping shall be analyzed on a case-by-case basis.

The fuel required to perform the station-keeping shall be accurately computed, since it influences the lifetime of the mission. Its quantity is dependent on the specific station-keeping orbital control to be implemented, and on the accuracy of the control (i.e., dimension of the station-keeping box). However, the station-keeping has a low impact on the other system budgets since it has a low frequency of operations and does not require complex computations. Differently from the attitude control, the execution of station-keeping commands is so infrequent that can be managed by a very basic on-board processor. Thus, if a spacecraft is capable to autonomously estimate its orbital status, it is very convenient to also implement autonomous station-keeping. Moreover, gravity has an exceptional short-term orbital control capacity. Then, if the orbital control system fails, the ground would easily determine that the satellite is slowly drifting from its assigned slot, and there would be enough time to fix the problem.



Attitude control system

Attitude control exploits the fundamental of feedback control systems, discussed in [Chapter 10 – Control](#), to be applied on the attitude dynamics.

Accordingly, the guidance and navigation functions use the fundamental principles discussed in the main chapters of this book in order to provide the reference pointing and the best attitude estimates. In this section, some attitude control methods are applied to solve the most common space-craft attitude problems.

Detumbling

Detumbling a spacecraft means to reduce its angular velocity to zero or to acceptable spin rate levels. Typically, detumbling can be triggered after the detachment from the deployer at the end of the launch, or in specific mission phases, or during emergency modes. For these reasons, the detumbling control should be as simple as possible, in order to be robust and suitable for any contingency operation.

Classic

The simplest way to stop the motion of a satellite is to apply a torque proportional to the angular velocity, opposite in sign. Let's assume to have the following candidate Lyapunov function:

$$L = \frac{1}{2} (\boldsymbol{\omega}^b)^T \mathbf{I} \boldsymbol{\omega}^b$$

Taking its derivative, substituting in the Euler equation presented in [Chapter 5](#) – Attitude Dynamics, and remembering that the dot product of a cross product is null, we have:

$$\dot{L} = (\boldsymbol{\omega}^b)^T \mathbf{t}^b$$

Hence, if we set $\mathbf{t}^b = -k\boldsymbol{\omega}^b$, we would have $\dot{L} \leq 0$ for any angular velocity, and the closed-loop system results to be stable. Of course, if we include external disturbances in the analysis, we would have convergence to a bounded area, provided that disturbances have a maximum known value.

The problem of this simple control law is that it requires a precise and unbiased angular speed estimation. Considering that most satellite embarks a gyroscope, this could be an issue only during off-nominal or initial detumbling operations, when it is not guaranteed that gyroscope's bias is correctly rejected, especially for small cost-effective satellites. This is the reason why another popular detumbling control strategy is instead commonly used in small LEO satellites: the B-dot [47].

B-dot

The B-dot algorithm is very popular in LEO missions that embark magnetic torquers. The name comes from the symbol \dot{B} representing the time derivative of the magnetic field measured by a magnetometer on-board the satellite. The reason why this control method is very important will be soon clear.

Recalling [Chapter 3](#) – The Space Environment, the magnetic field of the Earth, or other planets, can couple with currents flowing on-board the satellite producing a magnetic torque on the satellite itself. This is the working principle of magnetic torquers: $\mathbf{t}^b = \mathbf{m} \times \mathbf{B}^b = -\mathbf{B}^b \times \mathbf{m}$.

If we are capable to compute a magnetic dipole moment, \mathbf{m} , somehow related to the angular velocity of the satellite, we would achieve something quite close to the classic detumbling case. Let us then analyze the reason why \dot{B} is related to this:

$$\mathbf{B}^b = {}_b\mathbf{A}_i \mathbf{B}^i(\mathbf{x}).$$

Namely, the measured magnetic field depends on the local magnetic field direction and intensity, and on the spacecraft attitude. Note that the local magnetic field is dependent on the spacecraft position, \mathbf{x} , and thus it varies along the orbit as well. If we take the time derivative of the magnetic field vector, we would have:

$$\dot{\mathbf{B}}^b = {}_b\mathbf{A}_i \dot{\mathbf{B}}^i(\mathbf{x}) - \boldsymbol{\omega}^b \times {}_b\mathbf{A}_i \mathbf{B}^i(\mathbf{x}).$$

In detumbling conditions, the angular velocity of the spacecraft is sensibly high and we could neglect the variation of the local magnetic field due to the orbital motion:

$$\dot{\mathbf{B}}^b \approx -\boldsymbol{\omega}^b \times {}_b\mathbf{A}_i \mathbf{B}^i(\mathbf{x}) = -\boldsymbol{\omega}^b \times \mathbf{B}^b$$

Hence, as desired, $\dot{\mathbf{B}}^b$ is mainly function of $\boldsymbol{\omega}^b$, and we can implement a detumbling control law based on magnetic measurements only. Indeed, \mathbf{B}^b is the magnetic field sensed by the on-board magnetometers, and its derivative can be computed with a finite difference approach. If we substitute this result in the control magnetic dipole moment, to be actuated by magnetic torquers, we can prove convergence to zero velocity:

$$\mathbf{m} = -k \dot{\mathbf{B}}^b$$

Hence, using the previous candidate Lyapunov function:

$$\dot{L} = (\boldsymbol{\omega}^b)^T \mathbf{t}^b = (\boldsymbol{\omega}^b)^T \left(\mathbf{B}^b \times k \dot{\mathbf{B}}^b \right) = k (\boldsymbol{\omega}^b)^T (\mathbf{B}^b \times \mathbf{B}^b \times \boldsymbol{\omega}^b),$$

which leads to:

$$\dot{L} = k (\boldsymbol{\omega}^b)^T [\mathbf{B}_\times^b]^2 \boldsymbol{\omega}_b$$

In order to have $\dot{L} \leq 0$, we need $k > 0$. This is because $[\mathbf{B}_\times^b]^2 = [\mathbf{B}_\times^b] [\mathbf{B}_\times^b]$ is not a positive definite matrix. In fact, we can express it as:

$$[\mathbf{B}_\times^b]^2 = - \|\mathbf{B}^b\|^2 \mathbf{I}_3 + \mathbf{B}^b (\mathbf{B}^b)^T,$$

where the diagonal elements are the most relevant. Moreover, the matrix $[\mathbf{B}_\times^b]^2$ changes over time, as the satellite is moving along the orbit, but the sign won't change as it is proportional to the norm of the magnetic field.

The B-dot detumbling control law allows to reduce the absolute angular velocity down to a value of the same order of magnitude as the orbit rate. Moreover, it allows to achieve this result with only magnetic components (i.e., magnetometers and magnetorquers) and with a simple GNC architecture. This method has proven to be strongly reliable and effective, and it became a standard for LEO spacecraft with magnetic actuators. Since the derivative of the magnetic field is commonly numerically computed, a filter could be employed to reduce the noise levels. Moreover, the B-dot is often implemented as a bang–bang control law, as explained in Ref. [48]. Note that to avoid feedback from the torquers on the magnetometers, the computation of the magnetic field derivative should be done during periods in which the magnetorquers are not actuated. Thus, the sensing and the actuation operations shall be decoupled in time, as also suggested in Chapter 6 – Sensors. Fig. 14.5 shows an example Monte Carlo analysis, with 100 runs, of a B-dot detumbling phase for a satellite in Earth orbit.

One-axis pointing

In many mission modes, it is required to point one axis of the satellite in one direction. This can happen in both the simple Sun pointing mode and in other special cases. The scope of the single-axis pointing control mode is to point one axis toward a single target, meaning that we want one direction of the satellite to coincide with something we want to look at. This can be

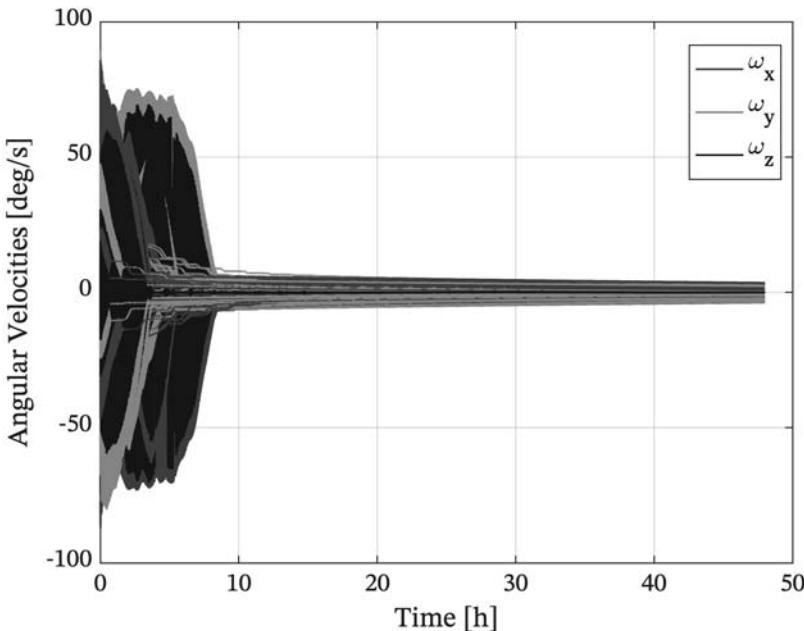


Figure 14.5 Monte Carlo analysis of B-dot detumbling.

either the Sun, the Earth, an astronomical object, another satellite, or anything else. The one-axis pointing guidance has been presented in [Chapter 8](#) – Guidance using quaternions; this section discusses the one-axis pointing attitude control exploiting the pointing error angle.

One-axis attitude control requires just two directions of the torque, \mathbf{t}^b , since the rotation around the target direction is not defined. We can define the pointing error angle as:

$$e = \cos^{-1}(\hat{\mathbf{s}}^b \cdot \hat{\mathbf{p}}^b),$$

where $\hat{\mathbf{p}}^b$ is the direction in body frame of the target we want to point, and $\hat{\mathbf{s}}^b$ is the spacecraft axis we want to align with the target. Thus, we want to minimize such error by controlling the satellite attitude toward the condition in which $\hat{\mathbf{s}}^b$ is aligned with $\hat{\mathbf{p}}^b$.

Let us assume, for the sake of simplicity, a reduced dynamical model where the angular velocities are low (i.e., the nonlinear terms of Euler equations can be neglected), and the attitude is expressed through a small-angle approximation with respect to the principal axes of the satellite. This is equivalent to linearize the attitude dynamics with respect to the reference

target-pointing condition. To improve clarity, we also assume $\hat{\mathbf{s}}^b = \hat{\mathbf{x}}$, but any other generic axis in body frame would be suitable. We can then express the target pointing direction in the reference frame of the axis of the satellite, using the small-angle approximation [48]:

$$\hat{\mathbf{p}}^b = {}_b\mathbf{A}_i \hat{\mathbf{p}}^i = (\mathbf{I}_3 - [\delta\vartheta_{\times}]) \hat{\mathbf{p}}^i,$$

where $\delta\vartheta$ are the small error angles in the spacecraft body frame, with respect to the principal axes.

Hence:

$$\hat{\mathbf{s}}^b \cdot \hat{\mathbf{p}}^b = \hat{\mathbf{x}}^T (\mathbf{I}_3 - [\delta\vartheta_{\times}]) \hat{\mathbf{p}}^i = [1 \quad \vartheta_z \quad -\vartheta_y] \hat{\mathbf{p}}^i$$

From which:

$$e = \cos^{-1}([1 \quad \vartheta_z \quad -\vartheta_y] \hat{\mathbf{p}}^i).$$

The pointing error angle is minimized if ϑ_y and ϑ_z go to zero, confirming our assumption that only two axes need to be controlled in order to keep the pointing. The dynamical evolution of the reduced system will then be:

$$\begin{cases} I_y \ddot{\vartheta}_y = t_y \\ I_z \ddot{\vartheta}_z = t_z \end{cases}.$$

This is a second-order system, and we can then express the control torque as:

$$t_y = -k_{py}\vartheta_y - k_{dy}\dot{\vartheta}_y,$$

giving:

$$I_y \ddot{\vartheta}_y + k_{dy}\dot{\vartheta}_y + k_{py}\vartheta_y = 0.$$

which is a harmonic oscillator with well-known properties. Without $k_{dy}\dot{\vartheta}_y$, the system would be undamped and, thus, not stable; hence, the need for a derivative term is evident. Given the inertia of the system, the tuning of the proportional-derivative (PD) controller is straightforward.

Let us try to generalize more the concept now. Take the following candidate Lyapunov function:

$$L = \frac{1}{2}(\boldsymbol{\omega}^b)^T \mathbf{I} \boldsymbol{\omega}^b + \frac{1}{2}k_p e^2$$

where $k_p > 0$, and the error is instead taken as $e = 1 - \hat{\mathbf{s}}^b \cdot \hat{\mathbf{p}}^b$, in order to avoid complications due to trigonometry. Its derivative is:

$$\dot{e} = \hat{\mathbf{s}}^b \cdot (\boldsymbol{\omega}^b \times \hat{\mathbf{p}}^b) = \boldsymbol{\omega}^b \cdot (\hat{\mathbf{p}}^b \times \hat{\mathbf{s}}^b)$$

The derivative of the Lyapunov function is:

$$\begin{aligned}\dot{L} &= (\boldsymbol{\omega}^b)^T \mathbf{t}^b + k_p e \dot{e} = (\boldsymbol{\omega}^b)^T \mathbf{t}^b + k_p e (\boldsymbol{\omega}^b)^T (\hat{\mathbf{p}}^b \times \hat{\mathbf{s}}^b) \\ &= (\boldsymbol{\omega}^b)^T (\mathbf{t}^b + k_p e (\hat{\mathbf{p}}^b \times \hat{\mathbf{s}}^b)).\end{aligned}$$

Let us define the cross product as a variable, closely linked to the error between directions:

$$\mathbf{h}^b = \hat{\mathbf{p}}^b \times \hat{\mathbf{s}}^b$$

Then, to achieve asymptotical stability ($\dot{L} \leq 0$), we would need to have:

$$\mathbf{t}^b = -k_d \boldsymbol{\omega}^b - k_p e \mathbf{h}^b.$$

This requires the knowledge of $\boldsymbol{\omega}^b$, which is not always available. There are a few options to be used, for example, we could use an approximation simply using the derivative of \mathbf{h}^b , which is closely related to the angular velocity. If we wish to be more rigorous, we could use the following control law:

$$\mathbf{t}^b = -k_d \mathbf{h}^b \times \dot{\mathbf{h}}^b - k_p e \mathbf{h}^b.$$

Substituting into the derivative of the Lyapunov function:

$$\dot{L} = k_d (\boldsymbol{\omega}^b)^T \left(-\mathbf{h}^b \times \dot{\mathbf{h}}^b \right)$$

Since $\dot{\mathbf{h}}^b = \boldsymbol{\omega}^b \times \mathbf{h}^b = -\mathbf{h}^b \times \boldsymbol{\omega}^b$, we have:

$$\dot{L} = k_d (\boldsymbol{\omega}^b)^T [\mathbf{h}^b \times]^2 \boldsymbol{\omega}^b.$$

That should satisfy the request for stability with $k_d > 0$. Instead, we could also have used $\hat{\mathbf{p}}^b \times \dot{\hat{\mathbf{p}}}^b$, as the implications would not change in theory. However, by definition, \mathbf{h}^b tends to zero as $\hat{\mathbf{p}}^b \rightarrow \hat{\mathbf{s}}^b$, while $[\hat{\mathbf{p}}^b \times]^2$ would not go to zero as it will approach $[\hat{\mathbf{s}}^b \times]^2$, which still does not have maximum rank.

The single-axis pointing control allows to steer the satellite in one direction, and as such can be quite robust and used, for example, in safe modes to point solar panels toward the Sun. In this case, the Sun direction measurements in body frame are needed in order to estimate the control torque required to align the prescribed body direction with the Sun. For small satellites, this might also be performed using magnetorquers, which would also require magnetic field estimation. However, the magnetic application of this control law would reduce the performance of the controller, since the command to the torquers depends on the cross product of the magnetic field, as will be discussed in the followings. This results in only one axis to be fully controllable and pointed.

Note that this control law is fully applicable to maneuver a spacecraft to a fixed attitude while driving the angular velocity to zero. It can be extended to also approximate the pointing control to quasiinertial reference directions. However, if the target pointing direction is moving, the controller shall be modified to include a reference time-varying attitude trajectory to be tracked. Further details can be found in many literature references discussing the attitude-tracking control [48].

Maximize secondary target

Single-axis pointing allows to point a single axis toward a target direction while keeping relative freedom to rotate around the target axis. This leaves open the possibility to make use of this degree of freedom to maximize the pointing of another axis. Clearly, it will not be possible to execute a perfect pointing with two axes if the angle between target directions is different from the angle of the body axes.

This implies that there is a way to minimize the error of a secondary target direction, but this will rarely be null. The control in this case can be easily derived, as one just need to project the secondary body axes direction and the secondary target direction on the primary target plane and, then, compute the relative angle. Once the angle between the two projected secondary directions is known, a simple additional PD controller can determine the secondary torque, which has to be aligned with the primary body axis direction. This additional secondary control action can be included during the primary orientation maneuver or after the primary target has been acquired. [Chapter 8](#) — Guidance presents a dedicated two-axis pointing guidance algorithm.

Three-axis pointing

In the cases, where the satellite attitude has to be fully controlled, without leaving anything to chance, it is recommended to exploit a full three-axis control technique, possibly using quaternions, although several alternative options are available. Theoretically, three-axis attitude control pointing is not possible, even with quaternions, since any kinematic parametrizations with three components have singularities and sets with more components (e.g., quaternions) have duality issues (e.g., quaternion unwinding). However, under a practical viewpoint, it is possible to derive stable three-axis control laws.

Let us consider a candidate Lyapunov function:

$$L = \frac{1}{4}(\boldsymbol{\omega}^b)^T \mathbf{I} \boldsymbol{\omega}^b + \gamma_q \frac{1}{2} (\boldsymbol{\delta q}_{13}^T \boldsymbol{\delta q}_{13}) + \gamma_q \frac{1}{2} (1 - \delta q_4)^2$$

where with $\boldsymbol{\delta q}_{1:3}$ and δq_4 we, respectively, consider the vector and the scalar components of the quaternion error, and $\gamma_q > 0$. The quaternion error is defined as the relative quaternion with respect to the target attitude quaternion, $\mathbf{q}_t = \begin{Bmatrix} \mathbf{q}_{1:3}^T & q_{4_t} \end{Bmatrix}^T$, and the estimated attitude, $\tilde{\mathbf{q}}$, following the quaternion multiplication rule:

$$\begin{Bmatrix} \boldsymbol{\delta q}_{13} \\ \delta q_4 \end{Bmatrix} = \begin{Bmatrix} \chi(\mathbf{q}_t) \tilde{\mathbf{q}} \\ \mathbf{q}_t^T \tilde{\mathbf{q}} \end{Bmatrix} = \begin{bmatrix} \mathbf{I}_3 q_{4_t} - [\mathbf{q}_{13_t} \times] & -\mathbf{q}_{13_t} \\ \mathbf{q}_{13_t}^T & q_{4_t} \end{bmatrix} \tilde{\mathbf{q}}$$

The evolution in time of the error is given by the following:

$$2 \dot{\boldsymbol{\delta q}_{13}} = \delta q_4 \boldsymbol{\omega}^b + [\boldsymbol{\delta q}_{13 \times}] \boldsymbol{\omega}^b$$

$$2 \dot{\delta q_4} = - \boldsymbol{\delta q}_{13}^T \boldsymbol{\omega}^b.$$

The derivative of the candidate Lyapunov function is:

$$\begin{aligned} \dot{L} &= \frac{1}{2}(\boldsymbol{\omega}^b)^T \mathbf{t}^b + \gamma_q \boldsymbol{\delta q}_{13}^T \boldsymbol{\delta q}_{13} - \gamma_q (1 - \delta q_4) \dot{\delta q_4} \\ &= \frac{1}{2} \left[(\boldsymbol{\omega}^b)^T \mathbf{t}^b + \gamma_q \boldsymbol{\delta q}_{13}^T (\delta q_4 + [\boldsymbol{\delta q}_{13 \times}]) \boldsymbol{\omega}^b \right. \\ &\quad \left. + \gamma_q (1 - \delta q_4) \boldsymbol{\delta q}_{13}^T \boldsymbol{\omega}^b \right] = \frac{1}{2} \left[(\boldsymbol{\omega}^b)^T \mathbf{t}^b + \gamma_q \boldsymbol{\delta q}_{13}^T \boldsymbol{\omega}^b \right]. \end{aligned}$$

If we use the following control torque:

$$\mathbf{t}^b = - \gamma_q \boldsymbol{\delta q}_{13} - \gamma_\omega \boldsymbol{\omega}^b$$

We would have:

$$\dot{L} = -\gamma_\omega \frac{1}{2} \|\boldsymbol{\omega}^b\|^2,$$

which is negative at the condition that $\gamma_\omega > 0$, guaranteeing the stability of the closed-loop system. However, this control law does not guarantee the shortest path to the final target state, which is an issue if $\delta q_4 < 0$ (i.e., *unwinding* phenomenon). Thus, a slightly modified control law is suggested to guarantee the shorter path to reach the desired equilibrium point:

$$\mathbf{t}^b = -\gamma_q \operatorname{sign}(\delta q_4) \mathbf{\delta q}_{13} - \gamma_\omega \boldsymbol{\omega}^b,$$

which is the PD control to achieve three-axis attitude stabilization.

This is certainly true for static inertial pointing $\boldsymbol{\omega}^b = 0$; however, this does not apply in the most general case, as convergence might be reduced or not guaranteed over the complete attitude phase space. Hence, the control law described before is valid for inertial pointing, while in the attitude-tracking case, we need to modify the analysis. Indeed, we need to impose the dynamics to follow a desired time-varying trajectory.

Let's modify the candidate Lyapunov function as:

$$L = \frac{1}{4} (\boldsymbol{\omega}^b - \boldsymbol{\omega}_t)^T \mathbf{I} (\boldsymbol{\omega}^b - \boldsymbol{\omega}_t) + \gamma_q \frac{1}{2} (\mathbf{\delta q}_{13}^T \mathbf{\delta q}_{13}) + \gamma_q \frac{1}{2} (1 - \delta q_4)^2,$$

where $\boldsymbol{\omega}_t$ is the target angular velocity in body frame. Under the assumption of $\dot{\boldsymbol{\omega}}_t \approx 0$ we obtain:

$$\dot{L} \approx \frac{1}{2} (\boldsymbol{\omega}^b - \boldsymbol{\omega}_t)^T \left[\mathbf{t}^b + \gamma_q \mathbf{\delta q}_{13} - \boldsymbol{\omega}^b \times \mathbf{I} \boldsymbol{\omega}^b \right]$$

Hence, the proper control law is:

$$\mathbf{t}^b = -\gamma_q \operatorname{sign}(\delta q_4) \mathbf{\delta q}_{13} - \gamma_\omega (\boldsymbol{\omega}^b - \boldsymbol{\omega}_t) + \boldsymbol{\omega}^b \times \mathbf{I} \boldsymbol{\omega}^b.$$

That is indeed a PD controller in terms of quaternions to achieve a tracking attitude three-axis pointing. The keen reader should have noticed that we could have neglected the nonlinear term of the Lyapunov function $\boldsymbol{\omega}_t^T (\boldsymbol{\omega}^b \times \mathbf{I} \boldsymbol{\omega}^b)$, whose influence disappears as $\boldsymbol{\omega}^b \rightarrow \boldsymbol{\omega}_t$. In the presented formulation, this term is anyway present; however, it might be relevant only for high angular rate, which is generally rare.

The term $\operatorname{sign}(\delta q_4) \mathbf{\delta q}_{13}$ avoids unwinding and instability, reminding that the direction cosine matrix (DCM) is quadratic in terms of quaternions,

such as a positive and a negative quaternion give the same attitude. However, it should be noted that the proportional term would be null at 180 degrees error (i.e., $\delta q_4 = 0$). Hence, to be rigorous in the control, one should include at least a modified sign function of δq_4 that is positive for $\delta q_4 > 0$ and negative otherwise.

Two loops

Readers familiar with other vehicle control techniques might easily see that there is also another way to deal with the three-axis attitude control problem, which is to divide the control function in two smaller concatenated loops.

If we only consider the attitude problem, we can write the Lyapunov function as:

$$L = \frac{1}{2}\gamma_q(\boldsymbol{\delta} \mathbf{q}_{t3}^T \boldsymbol{\delta} \mathbf{q}_{t3}) + \frac{1}{2}\gamma_q(1 - \delta q_4)^2 \rightarrow \dot{L} = \gamma_q \boldsymbol{\delta} \mathbf{q}_{t3}^T (\boldsymbol{\omega}^b - \boldsymbol{\omega}_t).$$

Thus, if we guarantee that $\boldsymbol{\omega}^b \rightarrow \boldsymbol{\omega}_t - \gamma_q \text{sign}(\delta q_4) \boldsymbol{\delta} \mathbf{q}_{1:3} = \boldsymbol{\omega}_r$, we guarantee convergence of the attitude loop. Hence, going back to the dynamics problem and taking this additional candidate Lyapunov function:

$$\dot{L} = \frac{1}{2}(\boldsymbol{\omega}^b - \boldsymbol{\omega}_r)^T \mathbf{I}(\boldsymbol{\omega}^b - \boldsymbol{\omega}_r),$$

we can derive it with respect to time obtaining:

$$\dot{L} = (\boldsymbol{\omega}^b - \boldsymbol{\omega}_r)^T \mathbf{I} \left(\dot{\boldsymbol{\omega}}^b - \dot{\boldsymbol{\omega}}_r \right) = (\boldsymbol{\omega}^b - \boldsymbol{\omega}_r)^T \left(\mathbf{t}^b - \boldsymbol{\omega}^b \times \mathbf{I} \boldsymbol{\omega}^b - \mathbf{I} \dot{\boldsymbol{\omega}}_r \right).$$

Thus, the following control law allows the dynamic loop to converge to the desired velocity:

$$\mathbf{t}^b = \boldsymbol{\omega}^b \times \mathbf{I} \boldsymbol{\omega}^b + \mathbf{I} \dot{\boldsymbol{\omega}}_r - \gamma_\omega (\boldsymbol{\omega}^b - \boldsymbol{\omega}_r)$$

This control law is quite general, and it can be used for any $\boldsymbol{\omega}_r$ profile. If we substitute the expression of $\dot{\boldsymbol{\omega}}_r$, we get:

$$\mathbf{t}^b = \boldsymbol{\omega}^b \times \mathbf{I} \boldsymbol{\omega}^b + \mathbf{I} \dot{\boldsymbol{\omega}}_r - \gamma_\omega (\boldsymbol{\omega}^b - \boldsymbol{\omega}_t) - \gamma_\omega \gamma_q \text{sign}(\delta q_4) \boldsymbol{\delta} \mathbf{q}_{t3},$$

which is pretty similar to the one we discussed in the previous section. In practical terms, $\dot{\boldsymbol{\omega}}_r$ could be computed via numerical derivation, instead of evaluating its elements, or, in some cases, it could be neglected similarly to $\boldsymbol{\omega}^b \times \mathbf{I} \boldsymbol{\omega}^b$.

As for what concerns the weights of the PD controller, it is straightforward to use a small-angle approximation to apply the classic control theory, which has been presented in [Chapter 10](#) – Control. If the two loops strategy is implemented, then it would be wise to set the cut-off frequency of the inner dynamic loop at least one decade higher than the one of the outer kinematic loop, in a way to reduce cross-effects.

Effects of disturbances

All these control examples have so far neglected the existence of disturbances, but these are always present. As previously mentioned, if the disturbances and the perturbations do not sum up over time, and their magnitude is reasonably smaller than the control action; then, the previous control laws should be sufficient. However, in many cases, the disturbances happen to have a continuous, nonnull action that would set the steady-state control error to a static nonzero value.

A common strategy in this case is to introduce in the controller an integral term based either on the angular velocity error or on the attitude error. To better visualize these two alternative possibilities, let us consider the two-loop scenario. If we place the integral term on the dynamic loop, it will reject only static disturbances given by the space environment; while, if we place it on the outer kinematic loop, it might also reduce the errors induced by the time-varying target reference signals.

Introducing integral terms in the control loop should be done with care, since they introduce delays that may reduce performances and control stability. This is especially true in those cases where the angular velocity of the target reference is high and time-varying. An example could be a ground station pointing case, where the satellite must quickly swipe to correctly point the antenna to guarantee communications. Moreover, since the integral action accumulates over time, proper saturations, resets, and antiwindup strategies must be implemented in the proportional-integral-derivative control algorithm.

Control with reaction wheels

Reaction wheels are used to control the spacecraft attitude by exchanging angular momentum with the satellite. Thus, they do not produce an external torque, since they are based on acceleration and deceleration of spinning rotors. Usually, when the nominal spin rate of the rotor is not maintained on a nonzero reference value, the actuator is denoted as *reaction*

wheel; alternatively, when a nonzero offset spin rate is desired, the actuator is called *inertia wheel*. The latter is commonly used when a dual-spin attitude stabilization is sought (cfr. [Chapter 5](#) – Attitude Dynamics). In some cases, the distinction between the two categories is not evident, and all these actuators can be classified as *reaction wheels*.

The general problem of control angular exchange devices can be modeled by writing the appropriate set of Euler equations, including the effects of the n actuators in the expression of the angular momentum:

$$\mathbf{h}^b = \mathbf{I} \boldsymbol{\omega}^b + \mathbf{h}_{\text{rotors}}^b = \mathbf{I} \boldsymbol{\omega}^b + \mathbf{R} \mathbf{h}_{\text{rotors}}^r,$$

where \mathbf{R} is the $3 \times n$ configuration matrix introduced in [Chapter 7](#) – Actuators, and $\mathbf{h}_{\text{rotors}}^r$ is a column vector with n elements representing the angular momentum of each reaction wheel.

Assuming the term $\mathbf{I} \dot{\boldsymbol{\omega}}^b$ is considering the presence of the rotors with no relative velocity, and that \mathbf{t}^b represents the disturbance torque, the attitude dynamics becomes:

$$\mathbf{I} \dot{\boldsymbol{\omega}}^b + \boldsymbol{\omega}^b \times \mathbf{I} \boldsymbol{\omega}^b + \dot{\mathbf{I}} \boldsymbol{\omega}^b + \dot{\mathbf{R}} \mathbf{h}_{\text{rotors}}^r + \mathbf{R} \dot{\mathbf{h}}_{\text{rotors}}^r + \boldsymbol{\omega}^b \times \mathbf{R} \mathbf{h}_{\text{rotors}}^r = \mathbf{t}^b,$$

and we need to include the equations of the relative dynamics of the rotors:

$$\dot{\mathbf{R}} \mathbf{h}_{\text{rotors}}^r + \mathbf{R} \dot{\mathbf{h}}_{\text{rotors}}^r = \mathbf{R} \dot{\mathbf{t}}_{\text{rotors}}^r,$$

where $\dot{\mathbf{t}}_{\text{rotors}}^r$ is a column vector with n elements representing the torques applied on each wheel.

The terms in the attitude dynamics equations depend on the specific actuator in use:

- $\dot{\mathbf{I}} \boldsymbol{\omega}^b$ is present only for CMGs, and it can be neglected if their inertia is small compared to the spacecraft one. Note that this term is also present when internal moving masses or inertia variations are present (e.g., fuel consumption, sloshing, robotic arms, etc.)
- $\dot{\mathbf{R}} \mathbf{h}_{\text{rotors}}^r$ exists only for CMGs, and it indicates the configuration variation in the spacecraft body frame. With reference to [Chapter 7](#) – Actuators, the configuration matrix variation has been expressed for the CMGs as $\mathbf{C} \dot{\boldsymbol{\theta}}_G$.
- $\mathbf{R} \dot{\mathbf{h}}_{\text{rotors}}^r$ is the spin rate variation term. It exists for reaction and inertia wheels, and it is usually zero for CMGs, since they commonly have constant spin rate.

- $\omega^b \times \mathbf{R} \mathbf{h}_{\text{rotors}}^r$ exists for any actuator, and it represents the rotor's angular momentum coupling with the angular rates of the spacecraft.

To solve the control problem, we can group these four terms in the control torque term as:

$$\mathbf{t}_{\text{control}}^b = -\dot{\mathbf{I}} \omega^b - \dot{\mathbf{R}} \mathbf{h}_{\text{rotors}}^r - \mathbf{R} \dot{\mathbf{h}}_{\text{rotors}}^r - \omega^b \times \mathbf{R} \mathbf{h}_{\text{rotors}}^r.$$

When the desired control torque to be commanded has been computed, the previous equation can be used to solve the effective actuation actions to be sent to the specific actuators in use. For instance, let's assume to have only reaction wheels with same spinning inertia, I_{rotors} , we can write:

$$\begin{cases} \mathbf{I} \dot{\omega}^b + \omega^b \times \mathbf{I} \omega^b = \mathbf{t}^b + \mathbf{t}_{\text{control}}^b \\ \mathbf{t}_{\text{control}}^b = -\mathbf{R} \dot{\mathbf{h}}_{\text{rotors}}^r - \omega^b \times \mathbf{R} \mathbf{h}_{\text{rotors}}^r \\ \dot{\mathbf{h}}_{\text{rotors}}^r = I_{\text{rotors}} \dot{\omega}_{\text{rotors}} = \mathbf{t}_{\text{rotors}}^r \end{cases}$$

Any control design technique discussed in [Chapter 10 – Control](#) can be used to calculate $\mathbf{t}_{\text{control}}^b$ after which $\mathbf{t}_{\text{rotors}}^r$ can be evaluated as:

$$\begin{cases} \mathbf{R} \dot{\mathbf{h}}_{\text{rotors}}^r = -\mathbf{t}_{\text{control}}^b - \omega^b \times \mathbf{R} \mathbf{h}_{\text{rotors}}^r \\ \dot{\mathbf{h}}_{\text{rotors}}^r = -\mathbf{R}^\dagger (\mathbf{t}_{\text{control}}^b + \omega^b \times \mathbf{R} \mathbf{h}_{\text{rotors}}^r) \end{cases},$$

where \mathbf{R}^\dagger is the pseudoinverse of \mathbf{R} , as discussed in [Chapter 7 – Actuators](#). Then, the command to be actuated on the wheels is simply available as:

$$\mathbf{t}_{\text{rotors}}^r = -\mathbf{R}^\dagger (\mathbf{t}_{\text{control}}^b + \omega^b \times \mathbf{R} \mathbf{h}_{\text{rotors}}^r),$$

where we neglected the loss torques on the rotors, or we assumed the reaction wheels to be equipped with a motor controller unit that is capable to achieve the desired $\mathbf{t}_{\text{rotors}}^r$ regardless of the internal friction and disturbances.

Desaturation

Looking at the equations for reaction wheels and, in general, for momentum exchange actuators, we may notice that they do not generate any external torque, but they exchange angular momentum with the spacecraft main body by means of internal torques, $\mathbf{t}_{\text{rotors}}^r$.

The torque $\mathbf{t}_{\text{rotors}}^r$ is, in general, provided by an electric motor that has its typical operational curve, which is a function of its angular speed. Despite

the variability of electric motor characteristic, it is commonly seen that there is a spin rate for which the available torque falls to zero. This is called saturation speed, and, when the rotor reaches this speed, the motor can provide no additional torque in the direction of the increasing spin rate. Hence, it requires a special desaturation mechanism to make it again fully operational.

In general, momentum exchange actuators are operated in order to keep them within the speed range for which the available maximum torque of the electric motor is approximately constant. In this way, the actuator can be modeled as a linear device if commanded by a current in the armature. If the control torque is more or less periodic, the actuator shall be sized in a way to be always maintained within its linear functioning regime. If the control torque is the combination of a periodic component and a secular component (i.e., to balance a nonsymmetric external perturbation), it will be inevitable to reach the saturation speed. In the case the secular component of disturbances is persistent, several desaturation maneuvers are required and they shall be scheduled within the sequence of routine operations. Moreover, the reader is invited to notice how angular exchange actuators are not suitable for detumbling or angular momentum dumping control. Indeed, the satellite spin rates would be accumulated in the rotors, leading to the wheel saturation.

Desaturation, or momentum dumping, or momentum unloading mechanisms must apply a net external torque to the satellite; therefore, they are based on thrusters or on magnetic torquers. Then, the desaturation control torque is designed to dump-out the excessive angular momentum stored in the rotors:

$$\mathbf{t}_{\text{control}}^b = -k \mathbf{h}_{\text{rotors}}^b.$$

This torque is actuated by the external torque devices, while the angular momentum exchange actuators are unloaded.

The desaturation control law can also be continuously active to dump excess momentum during the mission control modes. In this case, two cascade control loops are present: the main attitude stabilization loop and a lower priority loop with the task of desaturating the momentum exchange actuators. The former uses the momentum exchange devices, while their momentum is simultaneously regulated by the second control loop acting on the external torque actuators (i.e., magnetorquers, thrusters, etc.) [46].

Control with magnetorquers

Satellites in LEOs can make extensive use of magnetic actuators, coupling with the magnetic field, to give rise to an external torque. The advantage is their reduced mass, low costs, and potentially infinite control action. However, there are some drawbacks that limit their functionalities, as discussed in [Chapter 7](#) — Actuators. Indeed, these actuators can only work when strong magnetic fields are available, they cannot produce full three-axis stabilization, and they require magnetic measurements that can be influenced by their own actuation.

We can write the torque generated by such actuators as:

$$\mathbf{t}_{\text{MAG}} = \mathbf{m} \times \mathbf{B} = -[\mathbf{B}_\times] \mathbf{m}.$$

The goal would be to define a rule to command the magnetic dipole \mathbf{m} and achieve the required control torque $\mathbf{t}_{\text{control}}$. All the previous quantities are expressed in body frame, but the frame superscript has been omitted for conciseness. $[\mathbf{B}_\times]$ is the cross product matrix, which is singular and, thus, it cannot be inverted. This reflects the fact that it is not possible to provide three independent components of the magnetic control torque because of the cross-product operation.

A possible solution to this problem would be to compute the magnetic dipole as:

$$\mathbf{m} = \frac{1}{\|\mathbf{B}\|^2} [\mathbf{B}_\times]^2 \mathbf{t}_{\text{control}}$$

In fact, substituting in the control torque expression, we get:

$$\mathbf{t}_{\text{MAG}} = -\frac{1}{\|\mathbf{B}\|^2} [\mathbf{B}_\times]^2 \mathbf{t}_{\text{control}}$$

and the matrix $[\mathbf{B}_\times]^2$ can be rewritten in a way that would give:

$$\mathbf{t}_{\text{MAG}} = \mathbf{t}_{\text{control}} - \frac{\mathbf{B}\mathbf{B}^T}{\|\mathbf{B}\|^2} \mathbf{t}_{\text{control}}.$$

where the outer product matrix $\mathbf{B}\mathbf{B}^T$ divided twice by the norm projects the desired control torque on the plane orthogonal to \mathbf{B} . Thus, with this control method, the useless part of the desired control torque (i.e., the one that is in the direction of \mathbf{B}) is canceled out. Note that, in this way, the magnetic control is equal to the desired one only if the desired control, $\mathbf{t}_{\text{control}}$, is really orthogonal to \mathbf{B} . In general, if magnetorquers are the main control actuators of a satellite, it is suggested to create a control law without passing

by the general control torque definition. In such a way, the control law specializes for the magnetic actuators and the effectiveness of such a control is improved [49].

Let's assume a single-axis pointing with maximization of a secondary axis along a different axis of the spacecraft. In particular, we would like to align the nadir direction (i.e., $-\mathbf{r}$) with the x -axis of the principal body frame, and the velocity vector with the body y -axis. The control law is the PD one discussed for the single-axis pointing. Note that the main pointing direction is not inertially fixed and, thus, the derivative error is computed with respect to orbital angular rate, in order to maintain the x -axis pointing to the Earth. The secondary direction (i.e., \mathbf{v}) is generally not orthogonal to the main one, but as described before, the secondary control loop minimizes the error angle between the secondary directions in the plane orthogonal to the primary one. Fig. 14.6 reports an example of this single-axis pointing control with magnetic actuators. In particular, the spacecraft in the example is orbiting on a 600 km Sun-synchronous orbit, it has a mass of 1.5 kg, and uses three magnetic torquers with maximum available dipole of 0.5 Am^2 . The magnetic control is able to dissipate the initial angular rates and orient the primary axis in about 2 h. The steady-state error with respect to nadir is in the order of 10 degrees.

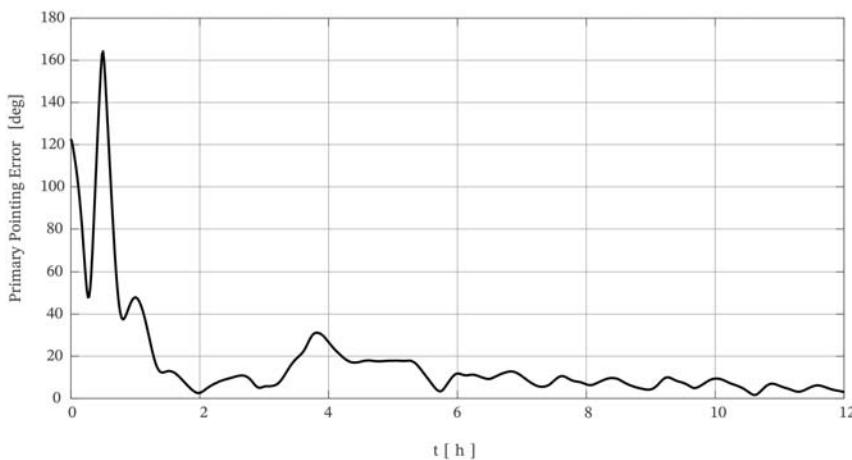


Figure 14.6 Single-axis magnetic control example.

Solar panels pointing

Solar panels pointing is a crucial task for an ACS, and it must be guaranteed even after multiple failures have occurred on-board. In these regards, it is possible to directly exploit the solar arrays as coarse Sun sensors to align the spacecraft to the Sun and achieve a single-axis Sun pointing, with the actuators only providing a torque along two axes in the plane orthogonal to the normal direction of the solar panels. The Sun direction estimation exploits the current/voltage sensing capacity of the electrical power subsystem to measure the Sun angle with respect to the arrays normal. It shall be noted that common spacecraft have the solar arrays deployed along a single body axis, with all the solar cells lying in a single plane. Hence, the complete Sun direction vector cannot be estimated from the electric measurements, and the Sun error axis is not known.

With these limiting assumptions, it is anyway possible to implement a GNC system that is capable to acquire the Sun and orient the solar arrays in a way to maximize the electric power generation. The estimation algorithms are based on the cosine law for the solar cells, also presented in [Chapter 6 — Sensors](#), and on a time-average for signal smoothing. The control loop is a classic PD law, which exploits a sequenced pulsed modulation to select the correct axis to rotate the spacecraft and align the solar arrays, despite the true error axis is not known. [Fig. 14.7](#) reports the Sun acquisition and panels alignment with this simple and reliable pulsed sequential PD control law. The complete description of an example of this ACS is discussed in Ref. [50].

Robust attitude control of a spacecraft with two rotating flexible solar arrays

In this applicative GNC example, it is shown how to model a flexible spacecraft with two rotating solar arrays and how to synthetize an ACS, robust to parameter uncertainties.

Let consider the spacecraft $\mathcal{S}\mathcal{C}$ in [Fig. 14.8](#). The spacecraft is composed of:

- the main body \mathcal{B} with its center of gravity B , its reference point O_b , and its body frame \mathcal{R}_b ,
- two symmetrical flexible solar arrays \mathcal{A}_1 and \mathcal{A}_2 connected to \mathcal{B} at the points P_1 and P_2 , respectively, through a revolute joint. A_i , O_{a_i} , and \mathcal{R}_{a_i} are the center of gravity, the reference point, and the body frame of \mathcal{A}_i ($i = 1, 2$).

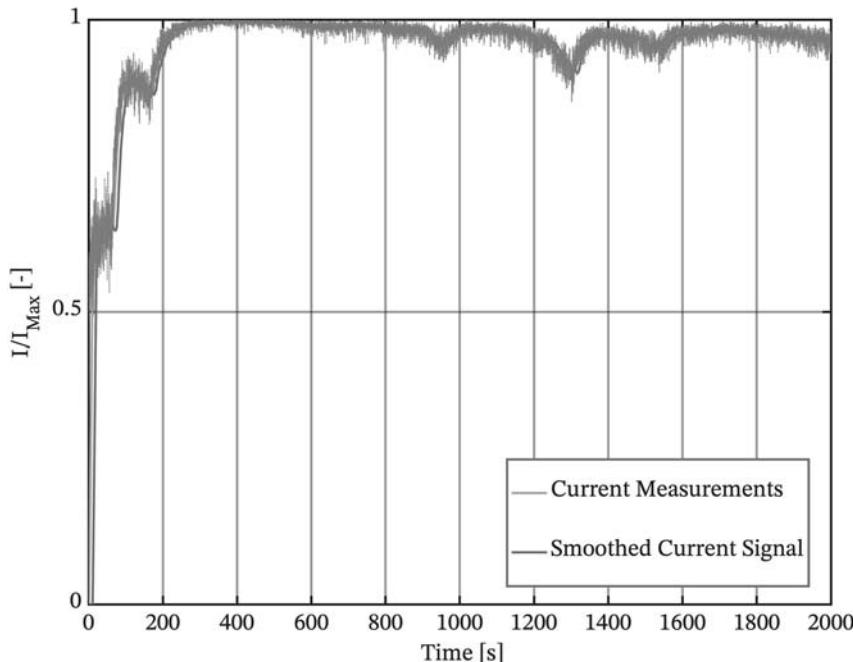


Figure 14.7 Solar panels pointing with pulsed sequential proportional-derivative control [50].

The angular configurations of the two solar panels are symmetrical: $\theta_1 = \theta$ and $\theta_2 = -\theta$. In Fig. 14.8, $\mathcal{R}_{a_i}(0)$ and $\mathcal{R}_{a_i}(\theta)$ represent two geometric configurations of \mathcal{R}_{a_i} for the nominal configuration ($\theta_i = 0$) and for a given angle θ_i .

$[\mathbf{D}_B^{\mathcal{LC}}]_{\mathcal{R}_b}^{-1}(s, \theta)$ is the model of the spacecraft at the center of gravity B of the main body \mathcal{B} and projected in the main body frame axes \mathcal{R}_b for a given angular configuration θ , that is the 6×6 transfer between:

- the resultant external wrench $[\mathbf{W}_{ext/\mathcal{B},B}]_{\mathcal{R}_b}$ (six components: three forces and three torques) applied to \mathcal{B} at the point B ,
- the dual vector of acceleration of point B $[\ddot{\mathbf{x}}_B]_{\mathcal{R}_b}$ (six components: three translations, three rotations).

Each revolute joint i adds in the model a single-input single-output (SISO) channel between:

- the torque u_i applied by \mathcal{B} to \mathcal{A}_i around \mathbf{x}_{a_i} inside the revolute joint,
- the relative acceleration $\delta\ddot{\theta}_i$ of \mathcal{A}_i w.r.t. \mathcal{B} around \mathbf{x}_{a_i} .

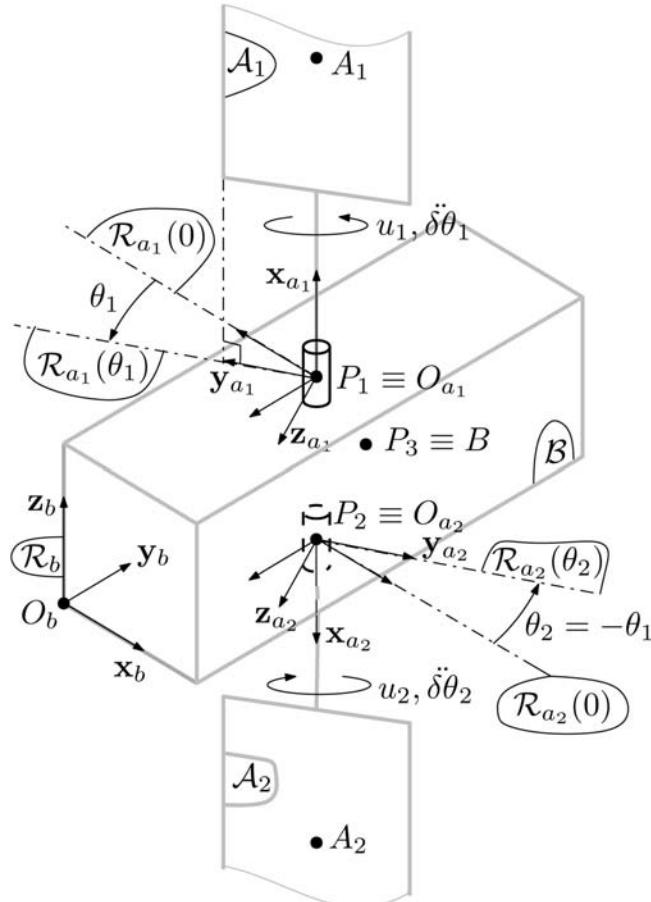


Figure 14.8 Spacecraft with two flexible rotating solar arrays.

To hold the angular configuration of \mathcal{A}_i w.r.t. \mathcal{B} at the value θ_i , the driving mechanism inside the revolute joint is modeled as a simple stiffness $k_i(\text{Nm}/\text{rad})$, a viscous friction $f_i(\text{Nms}/\text{rad})$, and an internal disturbing torque $p_i(\text{Nm})$:

$$u_i = -k_i \delta \theta_i - f_i \dot{\delta \theta}_i + p_i$$

Substructuring modeling

The spacecraft can be modeled in a multibody philosophy by assembling three different models corresponding to its building substructures. The approach we use is the two-input two-output Port (TITOP) framework [2,3].

Let us consider a flexible body \mathcal{L}_i as in Fig. 14.9 connected to a parent structure \mathcal{L}_{i-1} at the point P and to a child structure \mathcal{L}_{i+1} at the point C . The resulting TITOP model $\mathbf{D}_{PC}^{\mathcal{L}_i}(s)$, schematized in Fig. 14.9, is a 12×12 linear dynamic model function of the Laplace variable s whose inputs are:

- $\mathbf{W}_{\mathcal{L}_{i+1}/\mathcal{L}_i,C}$: the 6×1 wrench (forces and torques) applied by the body \mathcal{L}_{i+1} to \mathcal{L}_i at point C ;
- $\ddot{\mathbf{x}}_P$: the 6×1 inertial acceleration (linear and angular) imposed by the parent body \mathcal{L}_{i-1} at point P to \mathcal{L}_i ;
and the *conjugated* outputs are:
- $\ddot{\mathbf{x}}_C$: the 6×1 components of the inertial acceleration of point C ;
- $\mathbf{W}_{\mathcal{L}_i/\mathcal{L}_{i-1},P}$: the 6×1 wrench applied by \mathcal{L}_i to the parent structure \mathcal{L}_{i-1} at point P .

All these input/output variables are projected in the body frame.

An extension of this basic TITOP model can be easily done for bodies with multiple child nodes [4].

Many elementary structures (rigid bodies, beams, plates, etc.), complex flexible bodies (finite element models, liquid sloshing, etc.), and mechanisms (reaction wheels, solar array drive mechanisms, robotic arms, etc.) can be modeled in this approach. A user-friendly MATLAB/Simulink toolbox has been developed, the Satellite Dynamics Toolbox library [5,6] that

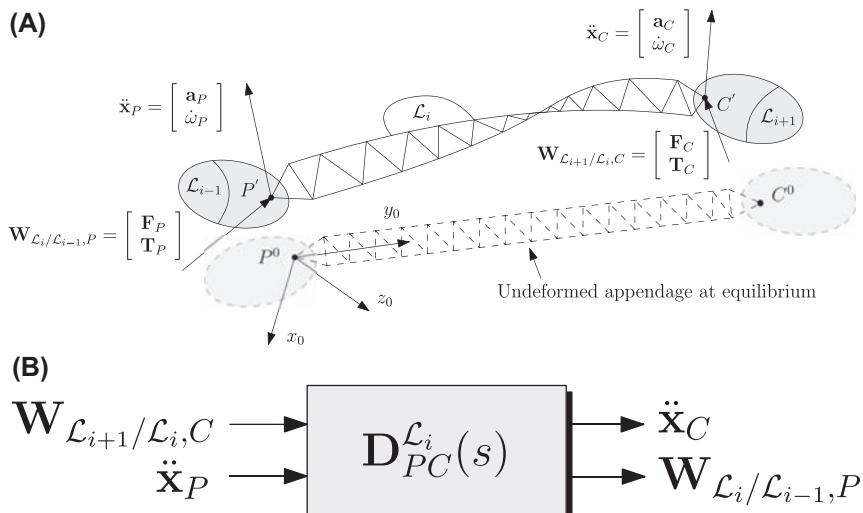


Figure 14.9 TITOP scheme and nomenclature for a generic flexible appendage \mathcal{L}_i .

integrates all these elements and allows an easy assembling of a complex multibody space system.

Main body

The inverse multiport TITOP model $\left[\mathbf{D}_{P_1, P_2, \dots, P_n}^{\mathcal{B}} \right]_{\mathcal{R}_b}^{-1}$ of a rigid body \mathcal{B} with several attachment points P_1, P_2, \dots, P_n and center of mass B is shown in Fig. 14.10.

Here the matrices $[\tau_{P_i B}]_{\mathcal{R}_b}$ represent the kinematic model between point P_i and B :

$$[\tau_{P_i B}]_{\mathcal{R}_b} = \begin{bmatrix} \mathbf{I}_3 & *(\overrightarrow{P_i B}) \\ 0_{3 \times 3} & \mathbf{I}_3 \end{bmatrix}$$

where $*(\overrightarrow{P_i B})$ is the skew-symmetric matrix associated with the vector $\overrightarrow{P_i B}$.

If $\overrightarrow{P_i B} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\mathcal{R}_i}$, where \mathcal{R}_i is an inertial frame, then:

$$*(*(\overrightarrow{P_i B}))_{\mathcal{R}_i} = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ y & x & 0 \end{bmatrix}_{\mathcal{R}_i}$$

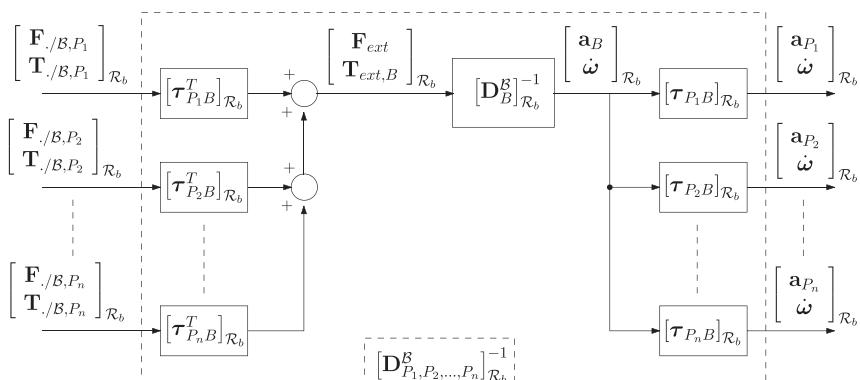


Figure 14.10 Multiport inverse TITOP model of a rigid body.

By referring to Fig. 14.11, the following data (all expressed in the body frame \mathcal{R}_b) are used for the current case study:

- geometry (in m): $\left[\overrightarrow{OP_1} \right]_{\mathcal{R}_b} = [0.4 \quad 1.4 \quad 0.5]^T$, $\left[\overrightarrow{OP_2} \right]_{\mathcal{R}_b} = [0.4 \quad 1.4 \quad 0]^T$, $\left[\overrightarrow{OB} \right]_{\mathcal{R}_b} = [0.35 \quad 1.5 \quad 0.5]^T$
- mass: $m^{\mathcal{B}} = 1000 \text{ kg}$
- 3×3 inertia tensor at B : $\left[\mathbf{J}_B^{\mathcal{B}} \right]_{\mathcal{R}_b} = \begin{bmatrix} 75 & 1 & 2 \\ 1 & 40 & -1 \\ 2 & -1 & 80 \end{bmatrix} (\text{kg m}^2)$

Flexible solar array with revolute joint

Let us consider the flexible body \mathcal{A} connected to a parent body \mathcal{P} through a revolute joint at a point P in Fig. 14.12. The 7×7 TITOP model of this system is composed of:

- The 6×6 channels from the absolute acceleration twist of the point P to the wrench applied by the body \mathcal{A} on the parent body, expressed in the parent body frame \mathcal{R}_p ,
- The 1×1 channel from the driving torque applied inside the revolute joint to the relative acceleration of the joint.

By referring to Fig. 14.12, the following parameters can be defined:

- \mathcal{R}_a : the body frame attached to \mathcal{A} ,
- \mathcal{R}_p : the body frame attached to \mathcal{P} ,
- r_a : the direction of the revolute joint axis and \tilde{r}_a a unitary vector along this direction,

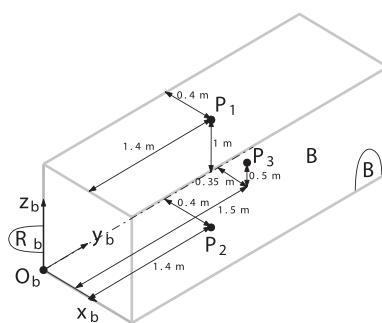


Figure 14.11 Main body geometry.

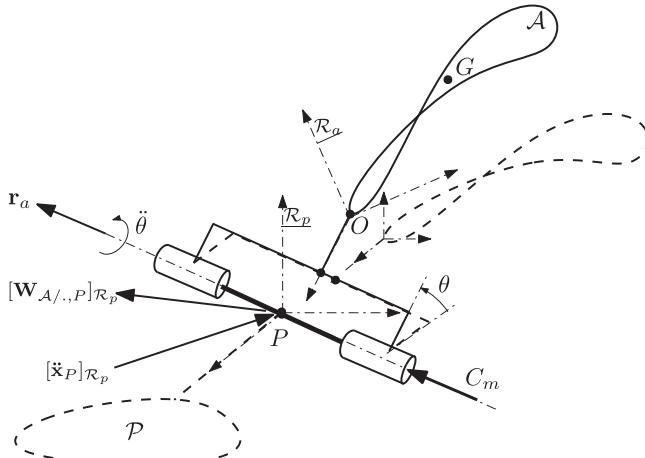


Figure 14.12 Schematic view of a flexible body with revolute joint.

- θ : the angular configuration of the revolute joint: \mathcal{R}_a is obtained from \mathcal{R}_p by a rotation of θ around r_a ,
- O and G: the reference point and the center of mass of body \mathcal{A} ,
- C_m : the driving torque which can be applied inside the revolute joint by an external driving system,
- $\ddot{\theta}$: the relative angular acceleration of the body side shaft with reference to the parent side shaft,
- $\left[\ddot{\mathbf{x}}_P \right]_{\mathcal{R}_p} = \begin{bmatrix} \mathbf{a}_P \\ \dot{\boldsymbol{\omega}} \end{bmatrix}_{\mathcal{R}_p}$: the absolute acceleration twist (6×1) of point P, expressed in \mathcal{R}_p ,
- $\left[\mathbf{W}_{\mathcal{A}/P} \right]_{\mathcal{R}_p} = \begin{bmatrix} \mathbf{F}_{\mathcal{A}/P} \\ \mathbf{W}_{\mathcal{A}/P} \end{bmatrix}_{\mathcal{R}_p}$: the wrench (6×1) applied by the body \mathcal{A} on the parent body, at point P, expressed in \mathcal{R}_p .

The TITOP dynamic model of the assembled system of flexible body and revolute joint is denoted as $[G_P^{\mathcal{A}+r}(s)]_{\mathcal{R}_p}^{-1}$, and it is represented by the block diagram in Fig. 14.13.

The detailed TITOP model is provided in Fig. 14.14. The following parameters of the flexible appendage are needed:

- Frequencies ω_i and damping ratio ζ_i ($i = 1, \dots, N$) of the N flexible modes

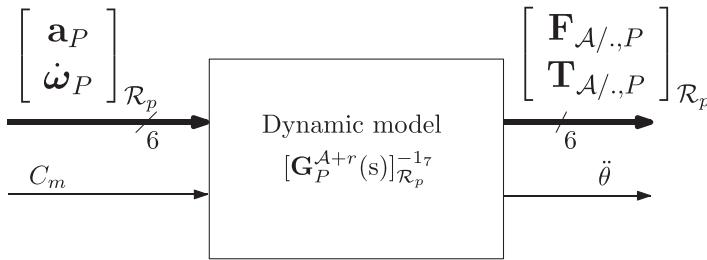


Figure 14.13 TITOP model of a flexible body with revolute joint.

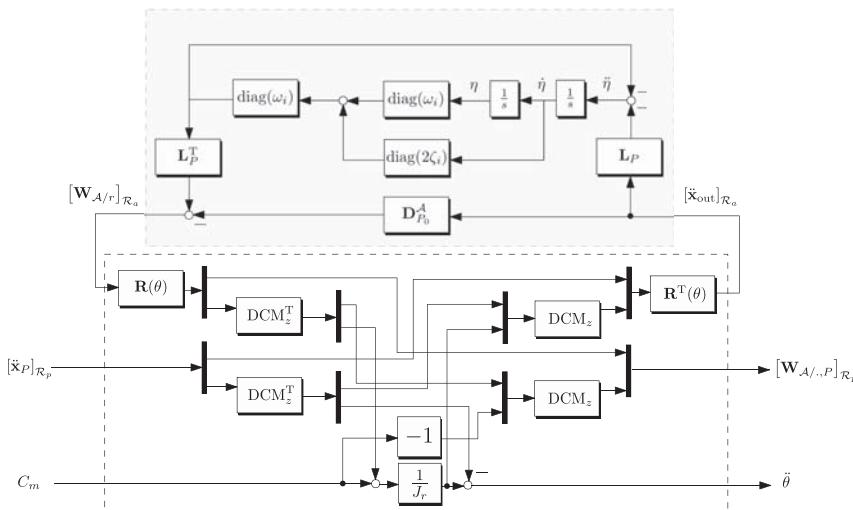


Figure 14.14 Internal dynamics of TITOP model of a flexible body with revolute joint.

- The $N \times 6$ matrix of modal participation factor \mathbf{L}_P with respect to point P
 - The residual mass $\mathbf{D}_{P_0}^{\mathcal{A}}$ of the body \mathcal{A} at point P
- For the revolute joint, the following parameters have to be defined:
- J_r : Inertia of the output shaft.
 - DCM_z : Direction cosine matrix which defines a new frame \mathcal{R}_z such that the z -third axis of the new frame \mathcal{R}_z is aligned with the revolute joint axis direction.
 - $\mathbf{R}(\theta)$: Direction cosine matrix between \mathcal{R}_a and \mathcal{R}_p . This matrix can be parametrized with the tangent of the quarter angle: $\tau = \tan\left(\frac{\theta}{4}\right)$ to derive a linear fractional transformation when θ is a varying parameter [7].

In the study case, the flexible appendages are the two identical solar arrays \mathcal{A}_1 and \mathcal{A}_2 . \mathcal{A}_1 is shown in Fig. 14.15 and its data are:

-

$$\left[\overrightarrow{O_{a_1} A_1} \right]_{\mathcal{R}_{a_1}} = [2.07 \quad 0 \quad 0]^T \text{ (m)}, \quad \left[\overrightarrow{O_{a_1} P_1} \right]_{\mathcal{R}_{a_1}} = [0 \quad 0 \quad 0]^T \text{ (m)}$$

- Mass: $m^{\mathcal{A}_1} = 43 \text{ (kg)}$

- Inertia tensor at A_1 : $\left[J_{A_1}^{\mathcal{A}_1} \right]_{\mathcal{R}_{a_1}} = \begin{bmatrix} 17 & 0 & 0 \\ 0 & 62 & 0 \\ 0 & 0 & 80 \end{bmatrix} \text{ (kg m}^2\text{)}$

- Vector of frequencies: $\omega = [5.6 \quad 19.3 \quad 35.4]^T \text{ rad/s}$

- Common damping ratio for the flexible modes: $\zeta = 0.005$

- Modal participation factors:

$$\begin{bmatrix} 0 & 0 & 12.5 & 0 \\ 0 & 0 & -5.1200 & -3.84 & -2.970 & 0 & 0 \\ 0 & 0 & 2.51 & 0 \end{bmatrix} \text{ (}\sqrt{\text{kg}}, \text{ m}\sqrt{\text{kg}}\text{)}$$

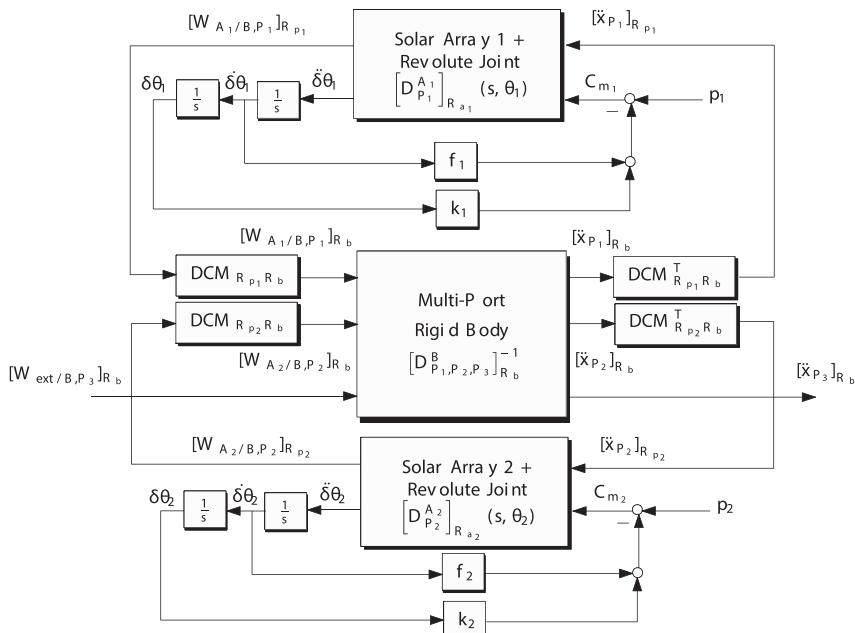


Figure 14.15 Assembled spacecraft in TITOP framework.

The revolute axis direction of \mathcal{A}_1 is $[\mathbf{r}_{a_1}]_{\mathcal{R}_{a_1}} = [1 \ 0 \ 0]^T$, while for \mathcal{A}_2 is $[\mathbf{r}_{a_2}]_{\mathcal{R}_{a_2}} = [-1 \ 0 \ 0]^T$.

Assembling of the whole spacecraft

The assembled spacecraft in TITOP framework is shown in Fig. 14.15. Note that the in-joint stiffness and friction k_i and f_i model a solar array drive mechanism.

For the numerical application: $k_1 = k_2 = 50000$ Nm/rad, $f_1 = f_2 = 100$ Nms/rad.

Moreover, the exogenous inputs p_1 and p_2 represent the harmonic disturbance induced by the driving mechanisms.

Fig. 14.16 shows the Bode diagram of the transfer function between the 3×1 vector of external torques $[\mathbf{W}_{ext/\mathcal{B}, P_3}]_{\mathcal{R}_b} \{4 : 6\}$ applied at the center of mass of the central body P_3 and the angular accelerations $[\ddot{\mathbf{x}}_{P_3}]_{\mathcal{R}_b} \{4 : 6\}$ of point P_3 . Notice that the system dynamics varies according to the parametric uncertainties listed in Table 14.5. In MATLAB, this parameter can be declared as *ureal*, thanks to the Robust Control Toolbox [8].

Robust attitude control synthesis

In this section, we show how to synthesize a robust ACS in order to meet some pointing and stability requirement in face of all system uncertainties.

The ACS architecture is shown in Fig. 14.17.

The basic idea is to use the combined measurements of a gyroscope (GYRO) and a star tracker (SST) to drive three control reaction wheels oriented in the three directions of the main body axes.

In Fig. 14.17, we can distinguish the following parameters:

- Spacecraft dynamics $[\mathbf{D}_{P_3}^{\mathcal{TC}}]_{\mathcal{R}_b}(s, \Delta, \tau_i)$ is the uncertain model of the spacecraft showed in Fig. 14.15, where the blocks Δ and τ_i include all uncertainties in Table 14.5.
- GYRO is the dynamical model of the gyro:

$$\text{GYRO} = \frac{400\pi}{s + 400} \mathbf{I}_3$$

- SST is the dynamical model of the star tracker:

$$\text{SST} = \frac{16\pi}{s + 16\pi} \mathbf{I}_3$$

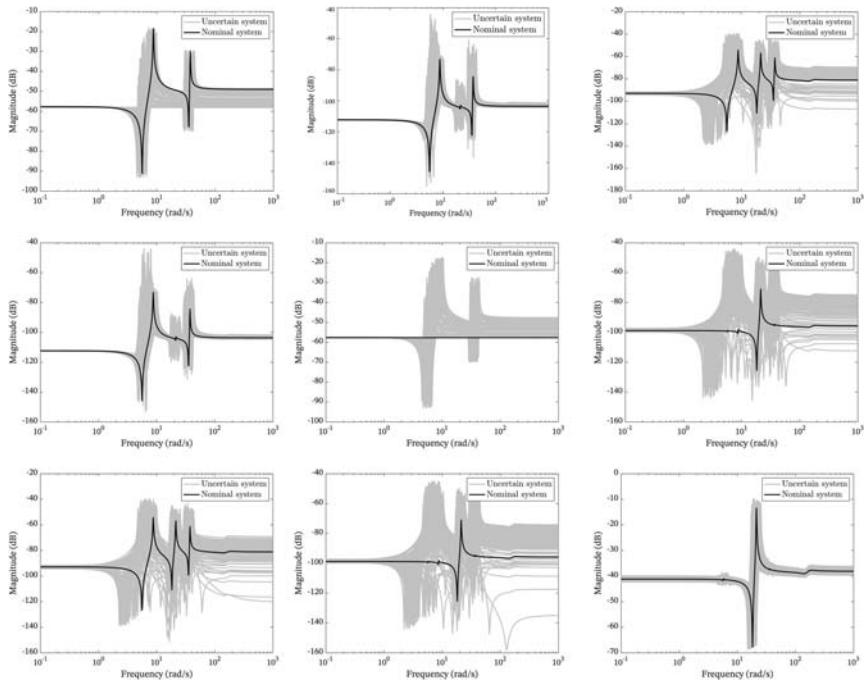


Figure 14.16 Bode diagram of the transfer $[W_{ext/B,P_b}]_{R_b}^{(4:6)}$ among the external torques and the angular acceleration of the spacecraft.

Table 14.5 Parametric uncertainties.

Parameter	Notation	Nominal value	Uncertainty
Main body mass	$m^{\mathcal{B}}$	1000 kg	$\pm 20\%$
Main body inertia x-axis	$[J_B^{\mathcal{B}}]_{R_b}^{xx}$	75 kg m ²	$\pm 20\%$
Main body inertia y-axis	$[J_B^{\mathcal{B}}]_{R_b}^{yy}$	40 kg m ²	$\pm 20\%$
Main body inertia z-axis	$[J_B^{\mathcal{B}}]_{R_b}^{zz}$	80 kg m ²	$\pm 20\%$
Solar arrays first mode frequency	$\omega_1^{\mathcal{A}_i}$	5.6 rad/s	$\pm 20\%$
Solar arrays second mode frequency	$\omega_2^{\mathcal{A}_i}$	19.3 rad/s	$\pm 20\%$
Solar arrays third mode frequency	$\omega_3^{\mathcal{A}_i}$	35.4 rad/s	$\pm 20\%$
Solar arrays rotations	$\tau_i = \tan \frac{\theta_i}{4}$	0	$[-1 \div 1]$

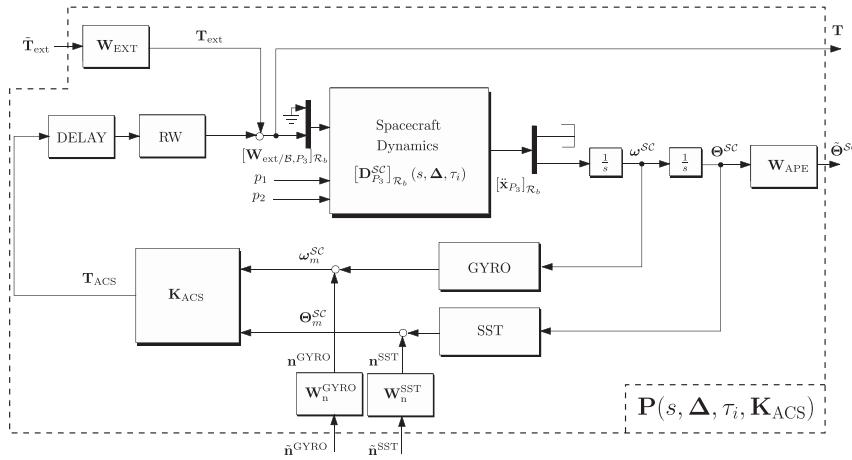


Figure 14.17 ACS architecture.

- Gyro noise, whose amplitude spectral density (ASD) is modeled with the weighting filter:

$$\mathbf{W}_n^{\text{GYRO}} = 1e^{-5} \mathbf{I}_3 \frac{\text{rad}}{\text{s}} / \sqrt{\text{Hz}}$$

- Star tracker noise, whose ASD is modeled with the weighting filter:

$$\mathbf{W}_n^{\text{SST}} = 1e^{-4} \mathbf{I}_3 \frac{\text{rad}}{\sqrt{\text{Hz}}}$$

- Reaction wheel dynamics:

$$\text{RW} = \frac{(200\pi)^2}{s^2 + 1.4 \cdot 200\pi + (200\pi)^2} \mathbf{I}_3$$

- ACS loop delay is modeled as a second-order Pade of time delay $\delta t_{AOCS} = 0.01$ s:

$$\text{DELAY} = \text{PADE}(\delta t_{AOCS}, 2) \mathbf{I}_3$$

- The weighting filter \mathbf{W}_{EXT} normalizes the input external torques to their expected upper bound:

$$\mathbf{W}_{\text{EXT}} = \begin{bmatrix} 0.03 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.02 \end{bmatrix} (\text{N m})$$

- The weighting filter \mathbf{W}_{APE} normalizes the spacecraft attitude error Θ^{SC} and overbounds the expected absolute performance error (APE):

$$\mathbf{W}_{\text{APE}} = \begin{bmatrix} \frac{0.01\pi}{180} & 0 & 0 \\ 0 & \frac{0.01\pi}{180} & 0 \\ 0 & 0 & \frac{0.05\pi}{180} \end{bmatrix}^{-1} \quad (\text{rad}^{-1})$$

The chosen ACS controller structure is a decentralized PD controller with a low-pass filter per axis.

The parameters to be tuned are thus the proportional gains K_p^x , K_p^y , and K_p^z , the derivative gains K_v^x , K_v^y , and K_v^z , and the cut-off frequencies of the three low-pass filters w^x , w^y , and w^z (Fig. 14.18).

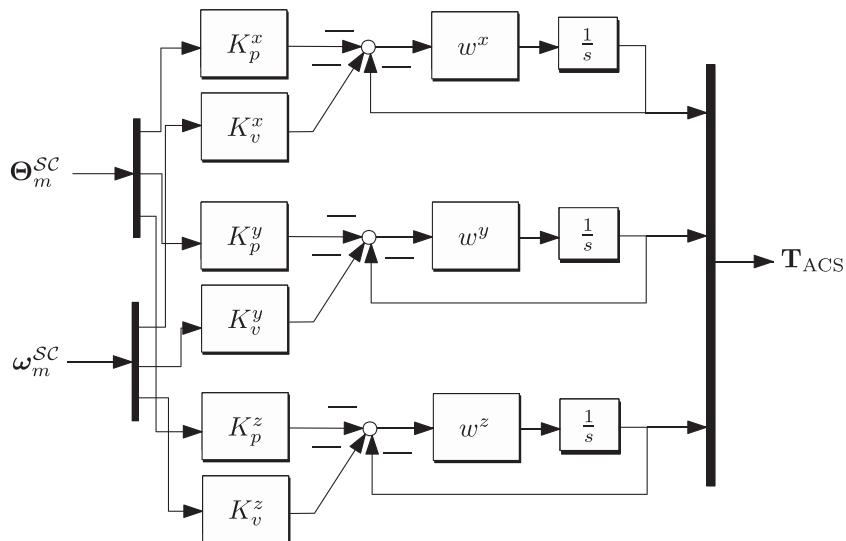


Figure 14.18 ACS control law K_ACS.

The optimization problem can be expressed in a robust control framework as:

$$\begin{aligned} \left\{ \hat{K}_p^x, \hat{K}_p^y, \hat{K}_p^z, \hat{K}_v^x, \hat{K}_v^y, \hat{K}_v^z, \hat{w}^x, \hat{w}^y, \hat{w}^z \right\} &= \min_{\mathbf{K}_{\text{ACS}}} J_c \\ &= \min_{\mathbf{K}_{\text{ACS}}} \max_{\Delta, \tau_i} \left\| \mathbf{P} \begin{bmatrix} \tilde{\mathbf{n}}^{\text{GYRO}} \\ \tilde{\mathbf{n}}^{\text{SST}} \\ \mathbf{n} \end{bmatrix} \rightarrow \mathbf{T} (s, \Delta, \tau_i, \mathbf{K}_{\text{ACS}}) \right\|_2 \\ \text{Such that: } &\begin{cases} \gamma_1 = \max_{\Delta, \tau_i} \left\| \mathbf{P}_{\tilde{\mathbf{T}}_{\text{ext}} \rightarrow \tilde{\Theta}} (\tilde{s}, \Delta, \tau_i, \mathbf{K}_{\text{ACS}}) \right\|_\infty < 1 \\ \gamma_2 = \max_{\Delta, \tau_i} \left\| \mathbf{P}_{\mathbf{T}_{\text{ext}} \rightarrow \mathbf{T}} (s, \Delta, \tau_i, \mathbf{K}_{\text{ACS}}) \right\|_\infty < 1.5 \end{cases} \end{aligned}$$

The optimization criterium $\min_{\mathbf{K}_{\text{ACS}}} J_c$ minimizes the \mathcal{H}_2 -norm (variance) from the normalized noise of both SST ($\tilde{\mathbf{n}}^{\text{SST}}$) and GYRO ($\tilde{\mathbf{n}}^{\text{GYRO}}$) to the input torque \mathbf{T} of the spacecraft in order to minimize the amplification of the sensor noise to the actuation effort.

The hard constraint γ_1 translates in the sense of \mathcal{H}_∞ -norm the requirement on the demanded APE pointing performance in face of the expected input external disturbance.

The second hard constraint γ_2 imposes finally an \mathcal{H}_∞ -norm of the sensitivity function smaller than 1.5, that ensures:

- A modulus margin bigger than $\frac{1}{\gamma_1} = 0.666$,
- A gain margin bigger than $\frac{\gamma_1}{\gamma_1 - 1} = 3$ (6 dB),
- A phase margin bigger than $2 \arcsin \frac{1}{2\gamma_1} = 38.9$ degree.

Notice that the optimization problem has also to guarantee an optimal solution valid for the whole set of parametric uncertainties.

The routine *systune* available in the MATLAB Robust Control Toolbox [8] and based on the nonsmooth optimization algorithm [9,10] allows to easily solve this optimization problem.

The results are provided in Table 14.6. We see that the variance of the noise has been reduced while satisfying both constraints on pointing requirement and sensitivity function by coping with all system uncertainties.

Table 14.6 ACS optimization result.

\hat{J}_c	$\hat{\gamma}_1$	$\hat{\gamma}_2$	\hat{K}_p^x	\hat{K}_p^y	\hat{K}_p^z	\hat{K}_v^x	\hat{K}_v^y	\hat{K}_v^z	\hat{w}^x	\hat{w}^y	\hat{w}^z
0.042	1.00	1.00	172.43	57.72	22.99	163.80	102.93	33.31	11.11	0.98	1.71

With *systune* we can also recover the worst-case configuration, that means the values of the set of uncertain parameters for which we obtain the worst performance. [Table 14.7](#) resumes their values:

Moreover, we can verify that the two constraints on pointing performance and sensitivity function are both verified by looking at the singular values of:

- The transfer $\mathbf{P}_{\tilde{T}_{\text{ext}} \rightarrow \tilde{\Theta}}(s, \Delta, \tau_i, \hat{\mathbf{K}}_{\text{ACS}})$ of the optimized plant for the pointing requirement in [Fig. 14.19](#).
- The transfer $\mathbf{P}_{\mathbf{T}_{\text{ext}} \rightarrow \mathbf{T}}(s, \Delta, \tau_i, \hat{\mathbf{K}}_{\text{ACS}})$ of the optimized plant for the sensitivity requirement in [Fig. 14.20](#).

We can notice that the \mathcal{H}_∞ -norm stays below 1 for the pointing requirement and below 1.5 for the sensitivity requirement.

We can finally verify the classical SISO stability margins by opening the closed loop in correspondence of the system input torque \mathbf{T} . [Fig. 14.21](#) shows the three Nichols plot for the diagonal terms of the transfer between input torque and spacecraft attitude angles.

One can check that the axis-per-axis stability margins (gain margins and phase margins) are largely better than the conservative lower bounds induced from sensitivity function requirement.

Table 14.7 Worst-case configuration.

Parameter	Nominal value	Uncertainty	Worst-case value
$m^{\mathcal{B}}$	1000 kg	$\pm 20\%$	800 kg
$[\mathbf{J}_B^{\mathcal{B}}]_{\mathcal{R}_b}^{xx}$	75 kg m ²	$\pm 20\%$	60 kg m ²
$[\mathbf{J}_B^{\mathcal{B}}]_{\mathcal{R}_b}^{yy}$	40 kg m ²	$\pm 20\%$	32 kg m ²
$[\mathbf{J}_B^{\mathcal{B}}]_{\mathcal{R}_b}^{zz}$	80 kg m ²	$\pm 20\%$	64 kg m ²
$\omega_1^{\mathcal{A}_i}$	5.6 rad/s	$\pm 20\%$	4.48 rad/s
$\omega_2^{\mathcal{A}_i}$	19.3 rad/s	$\pm 20\%$	23.16 rad/s
$\omega_3^{\mathcal{A}_i}$	35.4 rad/s	$\pm 20\%$	42.48 rad/s
$\tau_i = \tan \frac{\theta_i}{4}$	0	$[-1 \div 1]$	1

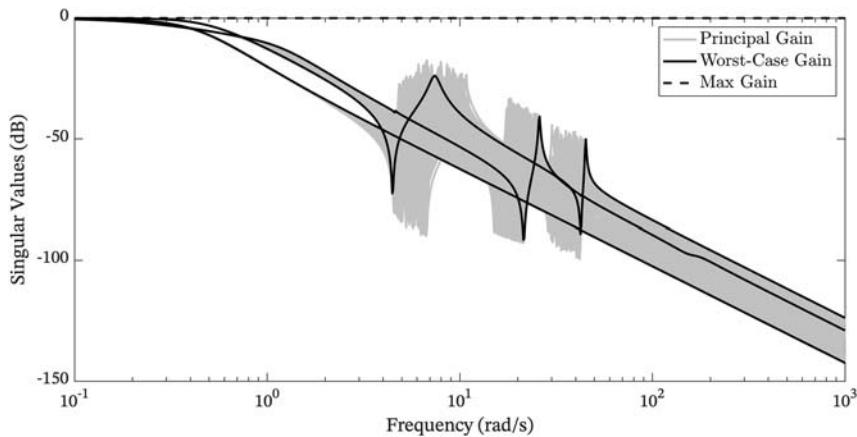


Figure 14.19 Pointing requirement verification.

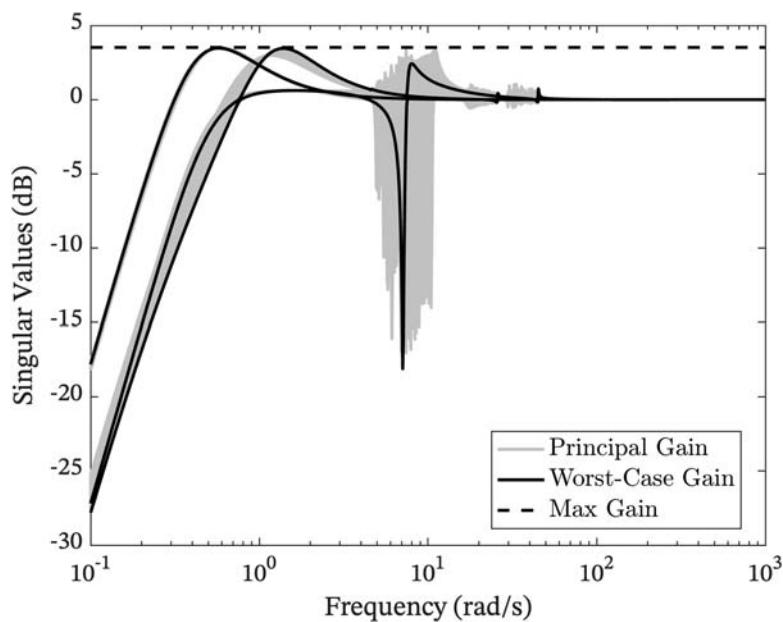


Figure 14.20 Sensitivity requirement verification.

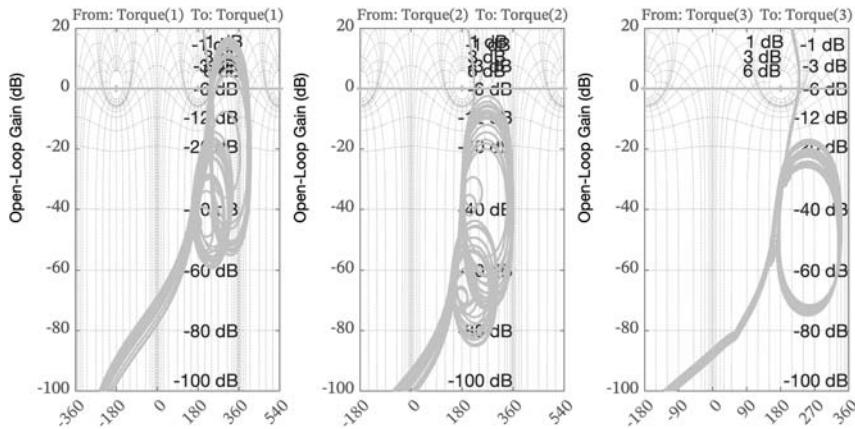


Figure 14.21 Nichols plots of the three spacecraft attitude axes.



Relative GNC

Relative GNC applications are dedicated to control the motion of a spacecraft relatively to other orbiting objects. The GNC functions are based on the relative dynamics equations presented in [Chapter 4](#) – Orbital Dynamics. The relative GNC is commonly applied to proximity operations, such as rendezvous and docking, fly around, on-orbit inspection, and on-orbit servicing; to formation flying and to any other application that impose any requirement on the relative states between two or more spacecraft.

Relative dynamics applications typically require real-time operations and a great autonomy level on the spacecraft. Thus, they cannot be assisted from ground and they commonly require all the GNC functions to be implemented on-board. For this reason, relative spacecraft dynamics was the first application of fully space-based GNC systems, and most of the developments for spacecraft autonomy were motivated by the requirements of proximity operations in space.

Guidance for relative and proximity maneuvers

The recent advances in spacecraft GNC unlocked daring missions involving orbital maneuvers in close proximity. Despite the first human-in-the-loop rendezvous missions date back to the 1960s, the enabling technology that now allows such missions is the development of autonomous GNC systems. Indeed, due to the close relative distances and the need for prompt reactions, impose the spacecraft to be capable of taking decisions rapidly and

autonomously. As discussed in [Chapter 9](#) – Navigation, the navigation requires dedicated sensors to perform relative and proximity operations. On the other hand, the control system typically relies on the same actuators used for other phases of the missions or at least the same type. Finally, as discussed in [Chapter 8](#) – Guidance, the guidance system can be tailored to derive optimal trajectories in a relative scenario. It is important to remark that, from the algorithms point of view, all the algorithms can be applied in the absolute scenario as in the relative one, providing that the constraints and capabilities of the system are the same for both.

In order to design effective maneuvers and proximity operations, it is critical to exploit the natural dynamics of the system. Hence, the design process is highly dependent on the trajectory design. In this section, we will go through some examples of potential strategies to design trajectories and algorithms for relative maneuvers with multiple agents.

Trajectory design and sensors selection

The trajectory design is highly dependent on the requirements of the mission. For instance, let us assume we want to fly around an object and perform images acquisition and perform optical navigation with respect to the target. Moreover, the design of the relative distances and attitude is dependent on the sensors adopted on-board, as discussed in [Chapter 9](#) – Navigation. With respect to the selection of vision-based sensors, a nonexhaustive set of specific criteria for monocular or stereo camera is:

- *Operational range.* The first consideration to be made concerns the size of the spacecraft, that constrains not only the available volume for the instrument but also more importantly the baseline between the camera’s axes, to be exploited for stereovision if selected. Such baseline b can be in the order of 30 cm (e.g., CubeSat), drastically reducing the operational range of the instrument. The disparity computed as difference of the two stereo frames is inversely proportional to the distance from the target. An estimation of the operational range of a stereo camera can be done computing the error on the target distance:

$$z = \frac{f \cdot b}{d}$$

where b is the baseline, f the focal length, and d the disparity. For example, with $b = 30$ cm, $f = 100$ mm, and a pixel of $10 \mu\text{m}$, a disparity of 1 pixel corresponds to an estimated distance of 3000 m, while a disparity of two

pixels corresponds to an estimated distance of 1500 m, causing a potential error of 1.5 km when at 3000 m from the target. In other words, in this case, the operational range of a stereo camera is very limited.

- *Resolution.* The resolution depends on the array size and distance from the target. For the stereo camera, there's the additional constraint of the Field of View (FoV) superposition.
- *Data type.* A stereo camera gives the possibility to directly retrieve 3D data, whereas the monocular camera delivers only 2D. It has to be noticed that data from monocular cameras can be elaborated on-ground and 3D information can be extracted as well with standard image processing algorithms.

To have full visibility of the target, relative motion of the chaser needs to be exploited. The relative trajectories must be passively safe; hence, natural motion shall be explored to achieve relative orbits, which do not threaten the safety of the image target phase. The imposed criteria to run the selection of the imaging relative trajectories are:

- *Passive safety* compliance that can be achieved.
- *Variance of geometrical coverage* with respect to target attitude motion which shall be minimized.
- *Sun-phase angle and FoV acquisition* during target pointing operations. FoV acquisition is intended as the objects, beside the target, that fall within the FoV (Sun, Earth, etc.).

One common approach for relative inspection is to exploit in-plane motion of the chaser around the target (cfr. [Chapter 4](#) – Orbital Dynamics). Using an in-plane motion, the relative orbit can be designed to be bounded and centered on the target. This makes trajectory design easier to handle. Nevertheless, constraining the trajectory to lay on a plane can be risky in terms of target coverage. If the target attitude motion is not known, planar motion may not be sufficient for the complete imaging accomplishment. For instance, a target spinning along the h-bar would prevent the chaser from imaging the faces directed in the h-bar direction. Moreover, the relative orbit crosses the v-bar twice in its natural motion, as shown in [Fig. 14.22](#). Such shortcoming may be solved by inserting an across-track component, i.e., out-of-plane motion, as shown in [Fig. 14.22](#). An additional step, which synthesizes the advantages of cross-track motion is to foresee passive trajectories that drift along the v-bar coupled with a cross-track motion component. In this way, the full coverage of the target can always be guaranteed, given the range of relative attitude during the inspection phase.

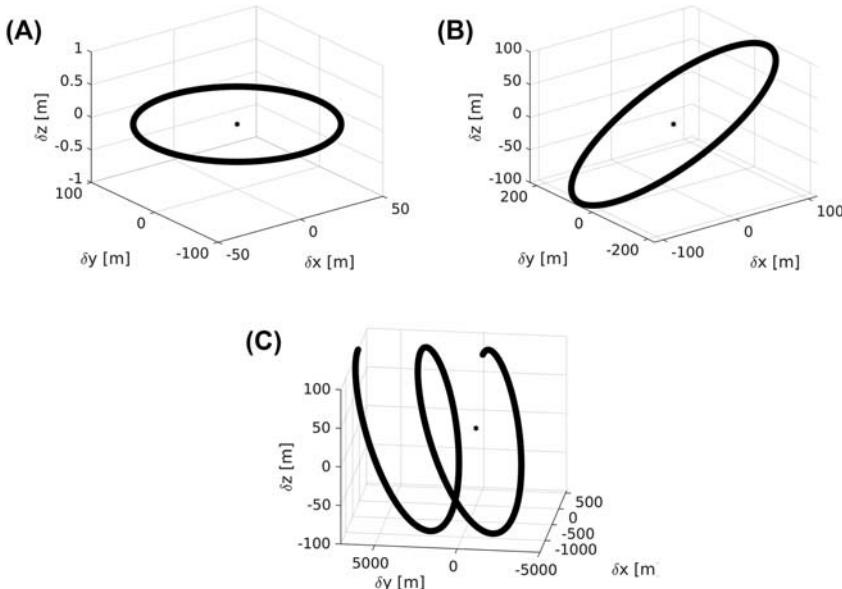


Figure 14.22 Target relative trajectory design. Planar $\delta e \neq 0$ (left), out-of-plane $\delta i \neq 0$ (middle), drifting $\delta a \neq 0$ (right).

Therefore, as far the relative trajectory possible strategies are concerned, three alternatives are valuable for consideration:

- *In-plane relative motion.* Two v-bar crossing weakening passive safety and not robust to resonant attitude motion.
- *Out-of-plane relative motion.* Increased target coverage but Earth in FOV when the trajectory is centered on the target.
- *Drifting relative motion.* Complete target coverage regardless of attitude motion and FOV acquisition and occultation limited.

As reported in [Chapter 4 – Orbital Dynamics](#), in the section about relative dynamics, the relative orbital elements (ROEs) can be used to design relative trajectories efficiently and effectively. As already mentioned, such parametrization allows to have a direct control of the relative trajectory geometry by using the invariant ROE. It is worth remarking that this approach is equivalent to the consolidated parametrization of absolute orbits: a specific orbit is identified by the orbital elements and not the inertial state.

For this reason, here a use case to design relative trajectories based on the definition of ROE is reported. Let us go back to the example where we want to fly around a target.

A planar fly around where an elliptical relative orbit is generated by the difference in eccentricity, i.e., $\delta e \neq 0$. A relative drift can be imposed by a difference in the semimajor axis, which determines the relative velocity. The planar motion may be restrictive in terms of imaging coverage.

Another solution is to introduce an out-of-plane motion by inserting a difference in the inclination between the chaser and the target. The relative drift is achieved as above. The differential J_2 perturbation is dependent on the true difference in inclination, Δi , which corresponds to δi_x , hence we would like to have the relative inclination vector only with δi_y component. Recalling section relative dynamics in [Chapter 4](#) – Orbital Dynamics, to make sure that passive safety is achieved, we impose that the maximum radial distance is achieved when the across-track distance vanishes. In other words, we impose that when the spacecraft crosses the target orbital plane, the radial distance is maximum. To achieve this, the orbits are design imposing δe and δi parallel.

A summary of the strategy for increasing the outcome of the proximity phase entails (please refer to [Chapter 4](#)):

- The planar trajectory ($\delta e \neq 0$) is too restrictive for imaging; hence, cross-track component ($\delta i \neq 0$) is necessary.
- To avoid differential J_2 -perturbation, we need to impose $\delta i_x = 0$.
- To ensure safety, we need $\delta e // \delta i$ thus constraining the $\delta e_x = 0$.
- To achieve relative drift $\delta a \neq 0$ determined by the desired relative drift rate $a\delta a = \frac{2d_{drift}}{3T_{drift}n}$.

Guidance and control strategies

The guidance and control strategy to be deployed on-board can be different. They are linked to the mission requirements and the capabilities that the spacecraft need to possess. In this section, few alternatives are presented and critically discussed.

Impulsive

The impulsive guidance has been described thoroughly in [Chapter 8](#) – Guidance. The reader is suggested to refer to that chapter for the mathematics of the algorithm. The idea is to calculate the required transfer trajectory between an initial state vector and a terminal state vector that are given at a fixed initial and final time. This means that the transfer time is fixed. Generally speaking, the transfer cost is larger for lower TOF. Hence, the mission design requires a trade-off between transfer time and the Δv cost.

Fig. 14.1–14.3 shows different transfer trajectory, for an example, initial condition of 2 km relative distance along each axis and initial velocity of $\delta v_0 = [-20, 20, -5] \frac{\text{m}}{\text{s}}$.

The transfer time is fixed. **Fig. 14.23** shows how the transfer trajectory changes as the fixed TOF increases. In particular, it is important to highlight that, as the transfer time becomes larger (e.g., five times the orbital period), the relative transfer exploits more the natural motion to reduce the cost. Indeed, a drifting trajectory is initiated when the transfer time is set to $t_f = 5T_{\text{orb}}$. In this example, the increased transfer time allows for a Δv cost saving of $\sim 10\%$ for $t_f = 0.5T_{\text{orb}}$ and $\sim 20\%$ for $t_f = 5T_{\text{orb}}$ with respect to $t_f = 0.05T_{\text{orb}}$.

The two-point impulsive maneuver is a fast algorithm to compute ideal trajectory. As discussed in [Chapter 8 – Guidance](#), it relies on the state-transition matrix of the employed dynamical model; hence, it is not robust to unmodeled perturbances. A possible solution is to recalculate the arc at a given frequency and employ a multitransfer strategy based on the actual estimated state on-board.

Artificial potential field

An alternative fast and autonomous strategy for on-board guidance and control is the artificial potential field (APF) technique [11–13]. The guidance strategy relies on artificial potential functions designed in the ROEs space in \mathbb{R}^6 .

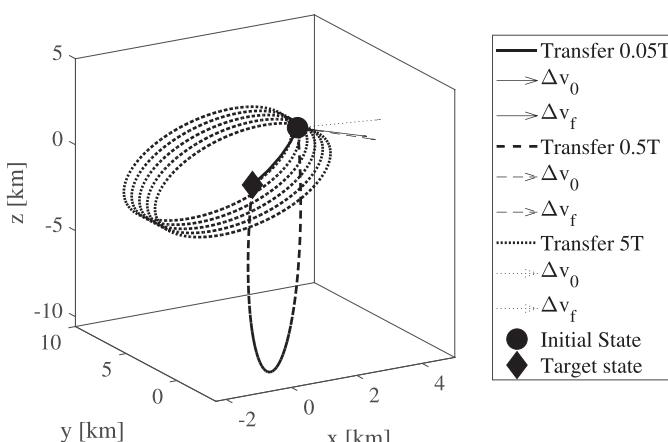


Figure 14.23 Two-point transfer for different transfer time.

The idea is to build a point-wise global potential based on the contribution of attractive and repulsive potential sources, namely the target relative orbits and any other satellite located in close neighboring areas. The attractive potential is directly expressed in terms of ROEs, those being a convenient way to express relative orbits geometry. Indeed, a set of ROEs uniquely define one particular formation configuration, i.e., space-craft relative positions. On the other hand, the natural way to express the vicinity between two satellites is using the Cartesian distance, expressed in the local-vertical-local-horizontal reference frame in this particular application. The output of the guidance, for each satellite, is what we call *guidance state* and indicate as $\delta\chi_g$. The guidance algorithm forces the following dynamics for each satellite i :

$$\dot{\delta\chi}_g = - \nabla \Phi_{glb}$$

where Φ_{glb} is the global potential:

$$\nabla \Phi_{glb} = \nabla \Phi_a + \nabla \Phi_r$$

where Φ_a is the attractive potential, whereas Φ_r is the repulsive one.

The reconfiguration objective is to drive the satellites to a predefined relative configuration, expressed in term of ROEs. The set of ROE to be achieved is called *reference state* and indicated as $\delta\chi_r$. The attractive contribution to the global potential is determined as:

$$\Phi_a(\delta\chi) = \frac{1}{2}\xi_a \left\| \delta\chi_g - \delta\chi_r \right\|^2$$

The gradient in the guidance ROE space is defined as:

$$\nabla_{\delta\chi_g} (\cdot) = \left(\frac{\partial}{\partial \delta a}, \frac{\partial}{\partial \delta \lambda}, \frac{\partial}{\partial \delta e_x}, \frac{\partial}{\partial \delta e_y}, \frac{\partial}{\partial \delta i_x}, \frac{\partial}{\partial \delta i_y} \right)_g$$

Consequently, the dynamic contribution given by the attractive potential is:

$$\nabla_{\delta\chi_g} = \xi_a \left(\delta\chi_g - \delta\chi_r \right)$$

Active collision avoidance

The repulsive potential is useful to calculate the trajectory in presence of other satellites, avoiding collision between agents. As previously stated, to achieve an efficient active collision avoidance maneuver, the potential is

best representative in terms of the Cartesian state \mathbf{x} in the Hill frame, where the metric distance is defined. Given two satellites, i and j , respectively, the repulsive potential to be computed for satellite i is defined as:

$$\Phi_{r_{ij}} = \begin{cases} \frac{1}{2}\xi_r e^{-\frac{d_{ij}^2}{\eta}} & \text{if } d_{ij} < d_{lim} \\ 0 & \text{if } d_{ij} > d_{lim} \end{cases}$$

where d_{lim} is the threshold distance beyond which the collision maneuver is not required, ξ_r , η are tuning coefficients. The gradient of the potential is calculated using the chain rule, which involves the coordinate transformation from Cartesian state \mathbf{x} to ROE $\delta\chi$:

$$\nabla_{\delta\chi_g} \Phi_{r_{ij}} = \nabla_{\mathbf{x}} \Phi_{r_{ij}} \cdot J_{\delta\chi}^{\mathbf{x}}$$

where $J_{\delta\chi}^{\mathbf{x}}$ is the Jacobian of the coordinate transformation; please refer to [Chapter 4](#) – Orbital Dynamics, in the section about relative dynamics. The gradient in the Cartesian space is defined as:

$$\nabla_{\mathbf{x}}(\cdot) = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)$$

Hence, the gradient of the repulsive potential between agents i and j , below the threshold, can be expressed as:

$$\nabla_{\delta\chi_g} \Phi_{r_{ij}} = -\frac{\xi_r}{\eta} e^{-\frac{d_{ij}^2}{\eta}} \cdot (\mathbf{x}_i - \mathbf{x}_j) \cdot J_{\delta\chi}^{\mathbf{x}}$$

The repulsive potential takes into account all the mutual distance between the neighboring spacecraft.

Tracking controller

The output of the guidance algorithm is a set of ROE, which may differ from the target reference ones. To guarantee that the forced guidance dynamics is followed, a feedback control law is employed. The control law is derived using the Lyapunov stability theorem. In the centralized architecture, the leader processes the guidance loop for the entire formation as well as the calculation of the control input for each satellite, which is then actuated by each agent. The current error between the desired guidance state and true state, for the entire formation, is:

$$e_{\delta\chi} = \delta\chi_g - \delta\chi$$

its temporal evolution can be described as:

$$\dot{e_{\delta\chi}} = \dot{\delta\chi_g} - \dot{\delta\chi} = -(\nabla\Phi_a + \nabla\Phi_r) - (A\delta\chi + Bu)$$

If we introduce the following positive semidefinite Lyapunov function:

$$V = \frac{1}{2} e_{\delta\chi}^T e_{\delta\chi} \rightarrow \dot{V} = e_{\delta\chi}^T \dot{e_{\delta\chi}}$$

$$\dot{V} = (\delta\chi_g - \delta\chi) \cdot [-(\nabla\Phi_a + \nabla\Phi_r + A\delta\chi + Bu)]$$

The control term can be solved to make the derivative of the Lyapunov function negative. The aim is to drive the derivative to the term $-e_{\delta\chi}^T e$, which is always negative. Hence, the following control law is derived:

$$u = \mathbf{B}^{-1} \left[(\delta\chi_g - \delta\chi) - (\nabla\Phi_a + \nabla\Phi_r) - A\delta\chi \right]$$

In this way, the derivative of the Lyapunov function is negative semidefinite, vanishing only when $\delta\chi = \delta\chi_r$, which is within the validity of the Lyapunov theorem.

Two main drawbacks can be identified for the APF algorithm:

- The APF may lead to instability due to the active-reactive nature of the algorithm. Nevertheless, user-defined parameters need to be tuned, tailored to the scenarios, to deliver acceptable maneuvers and control actions. Moreover, it is a continuous control law that may introduce operational constraints.
- The APF does not exploit in a predictive manner the knowledge of the dynamics. It is a real-time algorithm with no optimization capabilities. This may lead to excessive $\Delta\nu$, being intrinsically suboptimal. Furthermore, the transfer time is not fixed and cannot be estimated a priori.

An example maneuver using APF is reported in Fig. 14.24. The cost to perform such maneuver is comparable to the two-point impulsive maneuver for a transfer time longer than ~ 20 orbital periods. The continuous control action is shown in Fig. 14.25. One can clearly see that, due to the second-order differential equation that governs the guidance, the control is oscillatory at convergence.

Model Predictive Control

Model predictive control (MPC) is a guidance and control strategy that combines optimization and feedback control. This type of controller has been described in Chapter 10 – Control. Moreover, the example of a

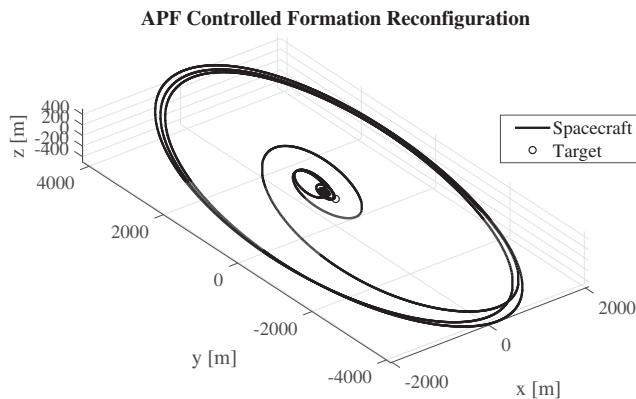


Figure 14.24 Example of APF relative maneuver from ~ 2000 m to along-track hold point.

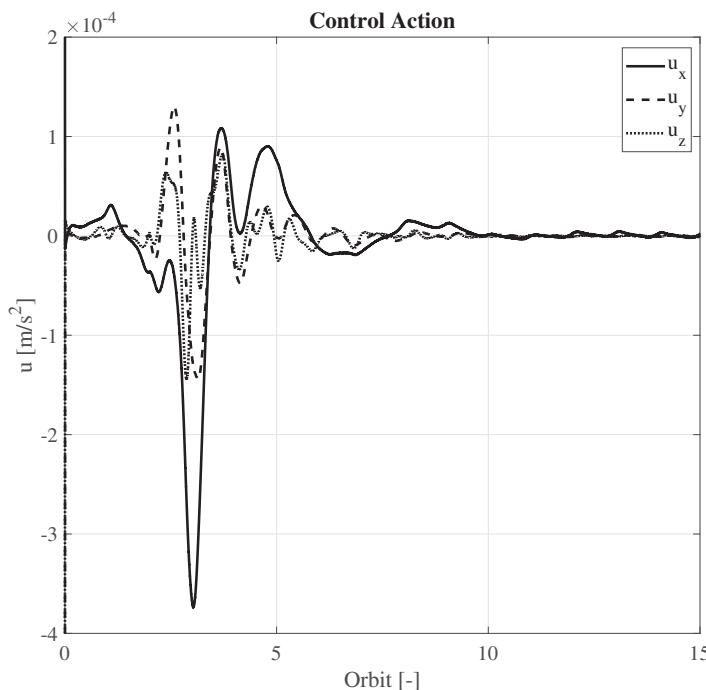


Figure 14.25 Control action for the example APF maneuver. The oscillatory behavior is visible at convergence.

controller for a relative maneuver has already been introduced in the same chapter, in case the optimal control problem could be casted into quadratic programming. In that example, we have seen how an optimal transfer can be solved by the MPC with dynamical and maximum thrust constraint. Here, we describe an example in which the problem has some nonlinear constraints. In particular, we will deal with the collision avoidance constraint, which is concave and quadratic.

The discretized cost function takes the form (cfr. Chapter 10 – Control):

$$\begin{aligned} \mathcal{J}(\mathbf{x}_k, \mathbf{u}_k) = & \left(\mathbf{x}_{k+N} - \tilde{\mathbf{x}}_k \right)^T \mathbf{Z} \left(\mathbf{x}_{k+N} - \tilde{\mathbf{x}}_k \right) \\ & + \sum_{i=1}^{N-1} \left(\mathbf{x}_{k+i} - \tilde{\mathbf{x}}_k \right)^T \mathbf{S} \left(\mathbf{x}_{k+i} - \tilde{\mathbf{x}}_k \right) + \sum_{i=0}^{N-1} \mathbf{u}_{k+i}^T \mathbf{R} \mathbf{u}_{k+i} \end{aligned}$$

where the final state $\mathbf{x}_f = \mathbf{x}(t_f) = \mathbf{x}_{k+N}$, with $t_f = N_s \cdot \Delta t_{MPC}$, in which the prediction horizon and the sampling time are introduced, namely N_s and Δt_{MPC} .

The collision avoidance constraint with respect to one obstacle can be expressed as:

$$1 - (\mathbf{x}_{k+i} - \mathbf{x}_{k+i}^o)^T \mathbf{C}^T \mathbf{P} \mathbf{C} (\mathbf{x}_{k+i} - \mathbf{x}_{k+i}^o) < 0, i = 1, \dots, N \quad (14.1)$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_3 \\ 0_3 \end{bmatrix}$$

where the superscript o refers to the *keep-out-zone* is an ellipsoid centered on the obstacle and expressed by its quadratic form with the positive semi-definite matrix \mathbf{P} . If the *keep-out-zone* is a sphere, the quadratic form is simply a symmetric matrix $\mathbf{P} = r^{-2} \mathbf{I}_3$, where r is the spherical safe region radius. The constraint is quadratic and nonconvex. The former characteristic prevents the problem to be recast into quadratic programming, which would significantly reduce the computational time. The latter turns the formulation into a nonconvex optimization. Convex programming is faster and guarantees the identification of global minima. For this reason, the following section presents the derivation to transform the collision avoidance constraint into a convex constraint [14,15]. Rearranging the term $\tilde{\mathbf{x}}_{o,k+i} = \mathbf{C}(\mathbf{x}_{k+i} - \mathbf{x}_{k+i}^o)$ in Eq. (14.1), the constraint can be written:

$$1 - \tilde{\mathbf{x}}_{o,k+i}^T \mathbf{P} \tilde{\mathbf{x}}_{o,k+i} < 0, i = 1, \dots, N$$

The idea is to approximate the concave set of feasible positions into a convex set, which contains the original set. In other words, the spherical *keep-out-zone* is approximated by a plane tangent to the sphere and perpendicular to the line of sight of the two agents. We assume to constrain the search space based on the nominal trajectory from the previous solution, here referred to as $\tilde{\mathbf{x}}_{o,k} \Big|_0$ with subscript 0. This is necessary to turn the constraint into an affine convex expression. These nominal values are assumed to be known and are not variables in the optimization. Therefore, the plane perpendicular to the vector $\tilde{\mathbf{x}}_{o,k} \Big|_0 = [\tilde{x} \Big|_0, \tilde{y} \Big|_0, \tilde{z} \Big|_0]$ and tangent to the sphere at instant k can be written as:

$$\begin{aligned} & \tilde{x}_{o,k} \Big|_0 \left(\tilde{x}_o - \frac{r}{\|\tilde{\mathbf{x}}_{o,k}\|_0} \tilde{x}_{o,k} \Big|_0 \right) + \tilde{y}_{o,k} \Big|_0 \left(\tilde{y}_o - \frac{r}{\|\tilde{\mathbf{x}}_{o,k}\|_0} \tilde{y}_{o,k} \Big|_0 \right) \\ & + \tilde{z}_{o,k} \Big|_0 \left(\tilde{z}_o - \frac{r}{\|\tilde{\mathbf{x}}_{o,k}\|_0} \tilde{z}_{o,k} \Big|_0 \right) = 0 \end{aligned} \quad (14.2)$$

The feasible position belongs to the semispace bounded by the identified plane. By reworking Eq. (14.2) and introducing the inequality indication, the compact form of the constraint at time step k is:

$$\tilde{\mathbf{x}}_{o,k} \Big|_0 \tilde{\mathbf{x}}_{o,k} \geq \mathbf{r} \|\tilde{\mathbf{x}}_{o,k}\|_0$$

where only $\tilde{\mathbf{x}}_{o,k}$ is the optimization variable. The nominal trajectory $\tilde{\mathbf{x}}_{o,k} \Big|_0$ represents an initial guess for the actual trajectory $\tilde{\mathbf{x}}_{o,k}$ and is used to convexify the collision-avoidance constraint. The closer the nominal trajectory is to the actual trajectory, the more accurate the convex program will be. The nominal trajectory plays an important role in the iterative methods for convex programming, which are beyond the scope of this book.

The optimal control problem reduces to a convex one:

$$\min_{\mathbf{u}} \mathcal{J}(\mathbf{x}_k, \mathbf{u}_k)$$

$$\text{subject to } \mathbf{x}_{k+i+1} = f(\mathbf{x}_{k+i}, \mathbf{u}_{k+i}), \quad i = 1, \dots, N_s$$

$$(\mathbf{x}_{k+i}|_0 - \mathbf{x}_{k+i}^o|_0)^T \mathbf{C}^T \mathbf{P} \mathbf{C} (\mathbf{x}_{k+i} - \mathbf{x}_{k+i}^o) \geq r \left\| \tilde{\mathbf{x}}_{o,k} \right\|_0$$

where again the nominal trajectory $\mathbf{x}_{k+i}|_0$ comes from the previous solution of the optimization step.

Optimal control

When dealing with relative dynamics and proximity operations, safety constraints, keep-out zones, and trajectory corridors usually impose the most stringent requirements to the GNC design. However, there is still left the possibility to optimize the relative GNC solution. The most common optimization goals are the time optimal, the fuel optimal, and the energy optimal GNC. The main differences exist in the formulation of the cost function.

In this section, an energy optimal guidance and control algorithm is presented to deal with the relative orbit and attitude dynamics of a chaser with respect to the target object:

$$\ddot{\mathbf{x}}_c = \ddot{\mathbf{x}} + \mathbf{a}_c$$

$$\delta \dot{\boldsymbol{\omega}}_c^C = \delta \dot{\boldsymbol{\omega}}^C + \boldsymbol{\alpha}_c,$$

where \mathbf{a}_c and $\boldsymbol{\alpha}_c$ are the translational and the rotational control acceleration vectors, which are added to the relative natural dynamics to get the controlled relative dynamics. The relative orbit and attitude dynamics are expressed by means of the equations presented in [Chapter 4](#) Orbital Dynamics and in [Chapter 5 – Attitude Dynamics](#).

The energy optimal rendezvous problem can be solved because the dynamics of the chaser is controlled by a control variable:

$$\mathbf{u} = \left[\frac{a_{C_x}}{a_{C_x\max}}, \frac{a_{C_y}}{a_{C_y\max}}, \frac{a_{C_z}}{a_{C_z\max}}, \frac{\alpha_{C_1}}{\alpha_{C_1\max}}, \frac{\alpha_{C_2}}{\alpha_{C_2\max}}, \frac{\alpha_{C_3}}{\alpha_{C_3\max}} \right]^T$$

which is representative of the 6 DoF normalized control accelerations, respectively, defined in the inertial frame and in the chaser body-fixed frame. The six control components are bounded: $-1 \leq u \leq 1$.

Many solutions of optimal control problems applied to spacecraft dynamics are based on indirect methods, relying on analytical relations. As discussed in the low-thrust trajectory design section, the conditions for optimality require the solution of a two-point BVP. It is well known that indirect methods ensure rapid convergence of good starting guesses, but most of the difficulties are related to the high sensitivity to the initial costates.

Indeed, the selection of a good initial guess for the costates is difficult and time consuming. For the applicative example discussed in this section, given the interest in exploiting these functions on-board, a more robust method is used: the optimal relative GNC problem is solved with direct methods, parametrizing only the control variable and converting the optimal control problem into an NLP problem, with a direct transcription process. Direct methods require often a large computation effort, but they are usually robust and can accommodate path constraints [56].

The solution of a generic NLP problem is a vector of n variables, \mathbf{p} , that minimizes a scalar objective function:

$$\min_{\mathbf{p}} F(\mathbf{p})$$

subject to m equality or inequality constraints:

$$\mathbf{b}_l \leq c(\mathbf{p}) \leq \mathbf{b}_u,$$

and bounds:

$$\mathbf{p}_l \leq \mathbf{p} \leq \mathbf{p}_u.$$

The equality constraints are obtained imposing $\mathbf{b}_l = \mathbf{b}_u$. With direct methods, the differential dynamic constraints of the indirect optimal relative GNC problem are converted into a set of algebraic constraints.

The optimality in terms of minimum energy control (i.e., minimum quadratic) is achieved defining the scalar cost function as:

$$F(\bar{\mathbf{p}}) = \frac{1}{2} \int_{t_0}^{t_f} \mathbf{u} |_{\bar{\mathbf{p}}}^T(t) \mathbf{u} |_{\bar{\mathbf{p}}}(t) dt.$$

The cost function is numerically integrated over the maneuver time from the control parametrization functions, knowing just the value of $\bar{\mathbf{p}}$. A constrained minimization algorithm can be then used to solve the NLP problem associated with the direct transcription of the optimal control. The initial guess for the parameters in the vector \mathbf{p} can be random, normally distributed within the bounds for the parameters. The complete relative trajectory path can be discretized in multiple arcs connected by patch points; for each of them, the energy optimal problem can be solved to find the best trajectory with the associated control vector.

The control action can be parametrized by means of polynomials up to the third degree and Fourier series up to the fourth order. The limitations in the degree of the expansions are motivated to limit the number of the design

parameters and, thus, the dimension of the NLP problem for potential on-board application. For example, the control parametrization with a second-degree polynomial for the translational control and with a fourth-order Fourier series for the rotation control results in:

$$\mathbf{a}_c(t) = \mathbf{a}_0 + \mathbf{a}_1 \left(\frac{t}{t_{ref}} \right) + \mathbf{a}_2 \left(\frac{t}{t_{ref}} \right)^2$$

$$\alpha_C(t) = \frac{\alpha_0}{2} + \sum_{k=1}^4 \left[\alpha_k \cos\left(k\tau \frac{t}{t_{ref}}\right) + \beta_k \sin\left(k\tau \frac{t}{t_{ref}}\right) \right],$$

where \mathbf{a}_i , α_i , β_i , and \mathbf{v} are 3×1 parameters vectors defined, respectively, in the inertial and in the chaser reference frame. These parameters compose the vector of unknown variables:

$$\mathbf{p} = \{\mathbf{a}_i, \alpha_i, \beta_i, \tau, t_f\},$$

to be found solving the problem in NLP. The reference time, t_{ref} , is needed to nondimensionalize the time, t , in the parametrized control functions. The choice of a reference time equal to the TOF is suggested for common applications. Note that alternative control parameterizations can be used, but the one discussed in this section guarantees a good compromise between robustness and fast convergence of the guidance and control algorithm for this applicative example.

The constraints are obtained from numerical integration of the controlled relative dynamics and are imposed in the optimal control problem. The keep-out zones and trajectory corridors can be enforced by means of penalties in the cost function or added as further constraints along the trajectory path points.

In any case, the relative states at the end of the relative trajectory have to satisfy the imposed boundary conditions at the final time.

Rendezvous in cislunar space

A modern applicative use case of the methods discussed in this section is the rendezvous in cislunar space, which has been proposed as ideal location to deal with space stations beyond Earth orbit. The on-orbit operations of a complex and, possibly, modular space system in lunar vicinity require autonomous relative GNC capabilities, to perform rendezvous and docking between uncrewed spacecraft in such peculiar space environment. Indeed, the cislunar space dynamics can be described exploiting a restricted three-

body problem modeling approach, including the gravitational attraction of the Earth and the Moon, and the perturbations due to the Sun [56,66].

This section presents some applicative result, for an example, rendezvous scenario with a passively cooperative target orbiting on a lunar Lagrangian point 2 (L2) Near Rectilinear Halo Orbit, with an orbital period of almost seven days and a periselene altitude in the order of 3000 km [56]. The terminal rendezvous operations can be macroscopically divided into three phases identified by the order-of-magnitude of the relative distances between chaser and target:

- Far-range rendezvous, from $\sim 10,000$ km to a 100 km distance.
- Close-range rendezvous, from 100 km to 1 km distance.
- Final approach rendezvous, from 1 km up to docking with the target.

During far-range rendezvous, the chaser is controlled in absolute position, with open-loop impulsive maneuvers for orbit control. In particular, the guidance and the control functions solve a Lambert-like problem to reach the desired hold-point before the close-range phase. The goal of the far-range rendezvous is to reach a final state relative to the target spacecraft. Hence, it is also exploited to accurately phase the two spacecraft before the close-range rendezvous. In this phase, the chaser attitude state is completely decoupled from the one of the targets, and it is controlled in absolute dynamics to satisfy the chaser system requirements.

The relative GNC starts to be effective from the close-range phase, which begins with a departure from a holding-point. The position of the chaser is controlled relatively to the one of the targets, and impulsive relative orbital control maneuvers are used. The attitude state is still decoupled from the target rotational motion; however, the orientation of the chaser is defined to satisfy relative navigation requirements.

The final approach rendezvous is entirely within the domain of a coupled 6 DoF relative GNC, which can be designed and optimized with the optimal control method discussed in the previous section. The low relative distances between chaser and target require continuous closed-loop forced translation for safety reasons. During the whole final approach rendezvous, the attitude control is entirely coupled with the position control in order to satisfy navigation (e.g., camera pointing) and docking (e.g., docking port alignment) requirements.

The relative navigation functions can be established on vision-based techniques. This is applicable after the close-range rendezvous phase, since short distances are needed to resolve the target in the sensor frame with the desired resolution to apply precise image processing techniques. The

navigation algorithm assumes that the only available data are provided by two cameras placed on the chaser. Its architecture is tightly coupled, since the measurements are directly processed by the navigation filter. The filter processes the features extracted by the two cameras to compute the relative target/chaser position and attitude. Since the observation model depends on both position and attitude of the target spacecraft, the navigation filter has to be coupled and nonlinear: an extended Kalman filter (EKF) can be used for this purpose [56]. Figs. 14.26–14.28 report some example results of the applicative rendezvous scenario in cislunar space.



On-board sensor processing

On-board sensors satisfy the primary sensing capacity to feed the whole GNC loop, starting from the navigation functions. The GNC system shall be thus capable to read and process their data, in order to have them ready and properly elaborated for the following GNC purposes. Fig. 14.29 shows a possible on-board sensor processing scheme, which is required to gather the sensor signals, interpret them according to structured data formats, and convert the signals into a value to be used by the GNC system [57].

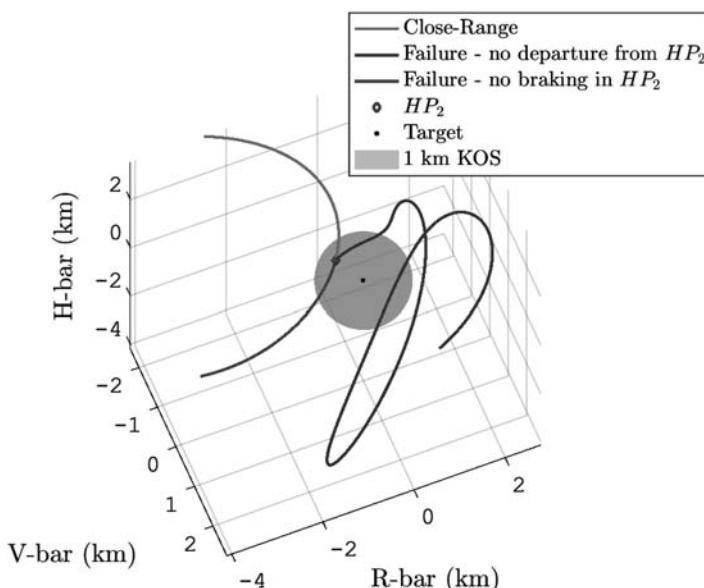


Figure 14.26 Close-range rendezvous final part: arrival at the keep-out sphere holding point (HP_2), with failures and passive safety enforcement. None of the trajectories enter inside the KOS.

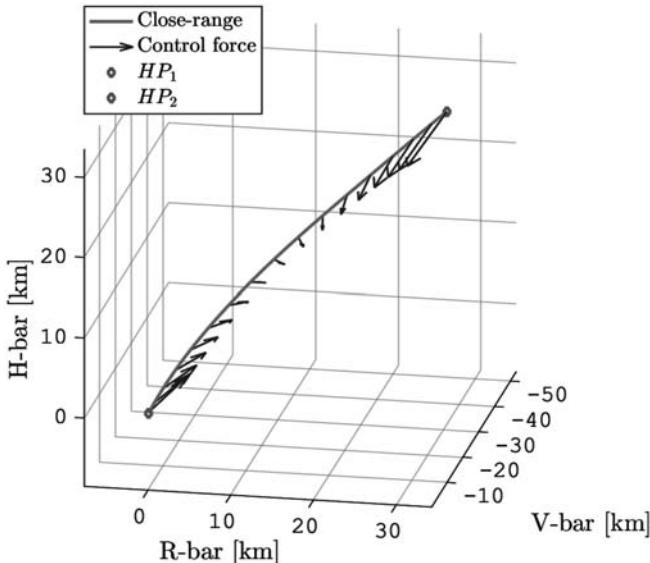


Figure 14.27 Close-range rendezvous trajectory with continuous thrust.

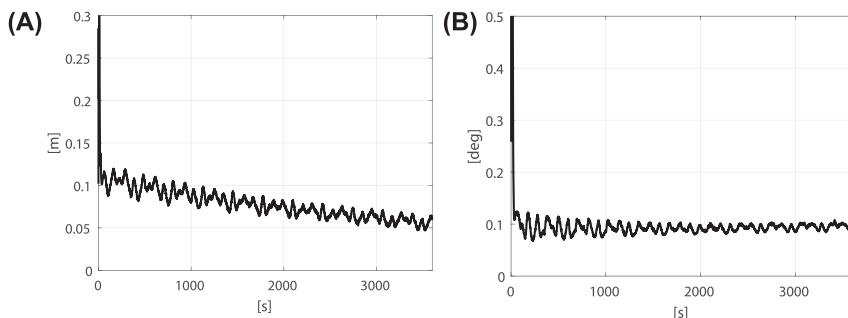


Figure 14.28 Navigation errors: final approach rendezvous at ~ 200 m from the target.
(A) Relative position navigation error. (B) Relative attitude navigation error.

The on-board sensor processing is directly interfaced with sensors outputs, which are provided as introduced in [Chapter 6 — Sensors](#). This GNC SW component includes the drivers to receive and correctly interpret the measurements from the HW peripheral components, the mathematical operations to scale and convert the measurements in the proper reference frame with correct dimensional units, the low-pass filters for those components without an embedded signal filtering, and the measurement correction functions. The measurements correction functions should contain all those

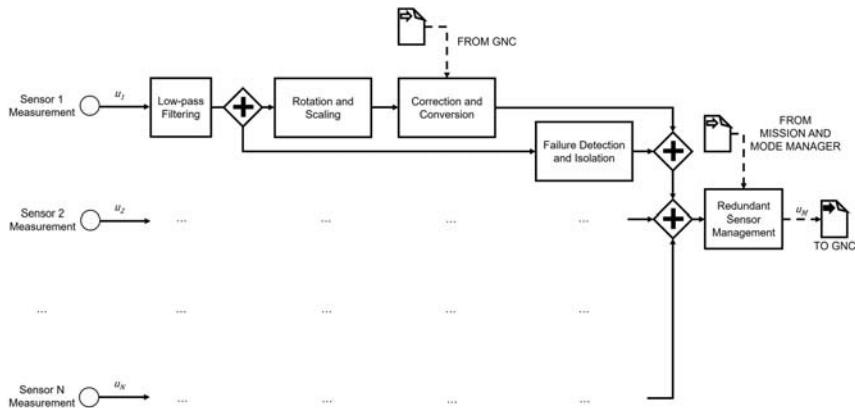


Figure 14.29 On-board sensor processing scheme [57].

operations needed to make the signal ready for the following estimation and GNC blocks. For instance, magnetometers can be compensated for the known dipole of the magnetic actuators, accelerometers are corrected for the gravitational and for the rotation induced accelerations, Sun sensors measurements are converted into Sun direction unit vectors and corrected for Earth's albedo, and similar operation are performed according to the specific sensor typology. Eventually, the measurements sent to the GNC can be processed by a redundant sensor manager, which selects, among the available sensor data, the best and not failed one in an active redundancy operation principles (cfr. Chapter 1 – Introduction).

The best measurement can be selected by analyzing the time running variance of the signals. In fact, the variance information of a given signal is directly dependent from the variance of the measured quantity, which is related to its dynamical evolution, and from the variance added by the measurement and acquisition chain. Since the measured quantity is the same for all the redundant sensors of a given typology, two redundant sensors have a different variance only because of distinct measurement and acquisition errors. In general, a higher variance is associated to larger stochastic errors in the measurements:

$$\begin{aligned}
 u_M = u_k &= \text{best}(u_1, \dots, u_n, \dots, u_N): \frac{1}{S} \sum_{i=1}^S \sigma_k(t_{S+1-i}) \\
 &< \frac{1}{S} \sum_{i=1}^S \sigma_j(t_{S+1-i}) \quad \forall j \neq k \in \{1, \dots, n, \dots, N\},
 \end{aligned}$$

where $\sigma_n(t_{S+1-i})$ is the running variance at the $(S + 1 - i)$ -th sample of time of the n -th sensor scalar measurement, which is computed on the last S samples as:

$$\begin{aligned}\sigma_n(t_s) &= \frac{1}{S} \sum_{i=1}^S u_n^2(t_{S+1-i}) - \left[\frac{1}{S} \sum_{i=1}^S u_n(t_{S+1-i}) \right]^2 \\ &= \frac{1}{S} \sum_{i=1}^S u_n^2(t_{S+1-i}) - \mu_n^2(t_S),\end{aligned}$$

where $\mu_n(t_S)$ is the running mean on the last S samples of time:

$$\mu_n(t_S) = \frac{1}{S} \sum_{i=1}^S u_n(t_{S+1-i}),$$

and t_S is the time associated with the S -th sample, since the sensor measurements are available at discrete samples of time. Hence, the best measurement is associated to the sensor with the lowest running mean of the variance on the last S samples of time.

The failed sensor measurements are identified by the sensor's failure detection and isolation (FDI) algorithms, and they are immediately excluded from the pool of available measurements to be selected by the redundant sensor manager.

Sensor failure detection, isolation, and recovery

FDI functions at component level typically operate on sensor signals after low-pass filtering, as shown in Fig. 14.29. Thus, the failure detection is performed on the raw filtered sensor measurements and can be based on statistical analysis on the incoming signal. These functions operate on the low-pass filtered data in order to have a uniform behavior among sensors with and without embedded low-pass filtering.

The FDI algorithm shall be designed and calibrated in order to detect the most common typologies of sensor faults: spike faults, erratic faults, drift faults, hardover/bias faults, data loss faults, and stuck faults [57], which have presented in Chapter 6 – Sensors. The complexity in detecting these faults is related to the dynamical evolution of the nominal sensor signals along a generic spacecraft dynamics. In fact, in general, none of the sensor outputs has a steady-state constant value, and the detection algorithms should be capable of detecting a variation that is not compatible with a feasible dynamical state of the spacecraft. For example, spacecraft nominal

rotations are bounded below maximum angular rates. Similarly, the accelerations are limited by the environment and by the maximum actuator forces and torques. Thus, there exists a maximum feasible time variation of the sensor signal that can be used to distinguish realistic measurements from faulty ones. The risk of marking real measurements as faulty ones (e.g., the spacecraft is really experiencing a nonnominal attitude dynamics) shall be minimized at system level, by exploiting the system-level failure detection functions (i.e., FDIR system), which are designed to compare more spacecraft components from different subsystems, and to detect dangerous conditions for the entire spacecraft. Note that the design approach should be as conservative as possible, preferring to detect false failures, rather than having failure propagation in the GNC system.

The failure detection algorithms at component level could be based on the running variance and running mean of the sensor signals. The running variances are used to detect the common sensor faults, while the running mean is applied to detect the presence of a valid sensor output. For this purpose, the sensor driver sets the sensor measurements equal to 0 if the sensor data are not received or received as corrupted packets.

A sensor is said to be nonfaulty if the running mean of the running variance is greater than zero and smaller than a given threshold, τ_n :

$$\min\left(\frac{1}{S} \sum_{i=1}^S \sigma_n(t_{S+1-i})\right) > 0 \wedge \max\left(\frac{1}{S} \sum_{i=1}^S \sigma_n(t_{S+1-i})\right) < \tau_n$$

The running mean of the variance is used to smooth the failure detection response and neglect confined errors in the signal. If the sensor measurement has multiple vector components, this relation is applied to each component. If these conditions are not respected, the failure is detected. For example, if the minimum component is equal to zero, the sensor has a stuck fault. If the maximum component is above the threshold, the other faults can be present: a high variance with fluctuations can indicate an erratic fault, short and repeated peaks in the variance can indicate spike faults, a single short peak in the variance can indicate a hardover fault. Drift faults are more difficult to be detected unless the drift is very fast. In this case, the variance has a significant constant increase with respect to nominal values. The variance threshold, τ_n , is specific for any sensor, and it shall be calibrated by ground and in-orbit testing. Note that the selection of the τ_n value is crucial to the proper operation of the FDI function.

A sensor is said to be operative if the square of the running is greater than zero. Thus, a sensor sends valid outputs if:

$$\min(\mu_n^2(t_S)) > 0.$$

If this condition is not respected for a prolonged time, the sensor is completely off. On the contrary, if the condition is not respected for short and close intervals of time, a data loss fault may be present.

If a failure is detected in a sensor, the redundant component manager is commanded to use the redundant sensors, and the outputs of the failed component are blocked. Then, isolation and recovery actions are executed in order to classify and identify the fault and, if possible, recover the faulty unit. Note that the switch to the redundant components is anyway a recover action at system level (cfr. [Chapter 11](#) – FDIR Development Approaches in Space Systems). Some example failure detection outputs are shown in [Fig. 14.30](#).

Autonomous on-board sensor calibration

Autonomous on-board sensor calibration is an operation that is used to estimate unknown error quantities to improve the GNC estimation performance. Indeed, the estimation and the knowledge of sensor bias errors, scale factors misalignment, nonorthogonality, and other low-frequency errors are crucial to achieve excellent accuracy in the navigation functions (cfr. [Chapter 6](#) – Sensors).

Autonomous on-board sensor calibration shall be performed since most of the low-frequency errors are not constant and fluctuate in time, due to thermal variations, ageing, and change in the environmental conditions. Thus, despite of great importance, the ground characterization and calibration of the GNC system is usually not enough to achieve the required navigation and control performance. The latter can properly estimate misalignment and nonorthogonality errors but fails in estimating unstable bias and other error drifts.

As said, a sensor is calibrated when all the deterministic, systematic, and the low-frequency drifting errors are removed from the available measurements. The estimation of sensor errors can be performed by exploiting different alternative methods; both batch approaches [58] and sequential filtering [59] can be used for sensor calibration. The traditional on-orbit calibration approach uses the EKF discussed in [Chapter 9](#) – Navigation, but different batch processes have been also developed to be periodically executed along the spacecraft timeline.

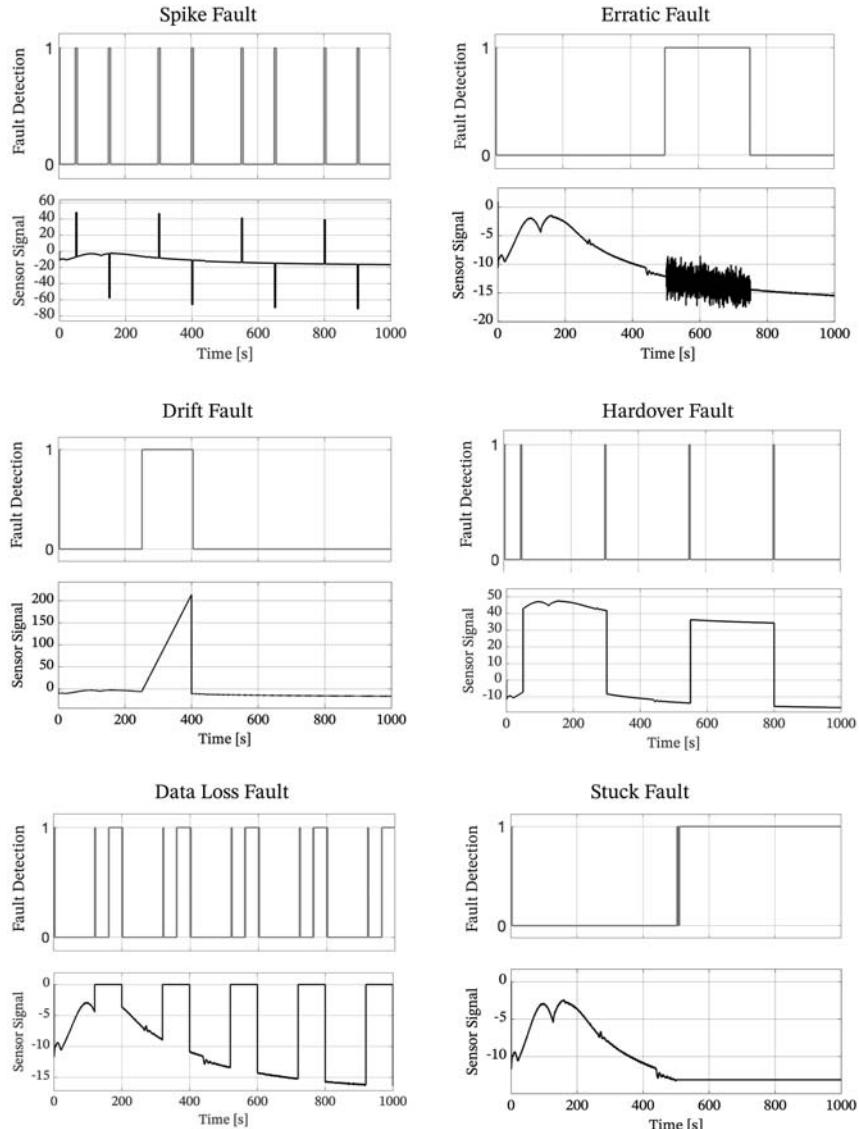


Figure 14.30 Sensor Faults and failure detection examples [57].

The sensors that are most commonly calibrated on-board are the inertial sensors, such as gyroscopes and accelerometers, and magnetometers. Other sensor types rely more on ground calibration to estimate misalignments, constant scale factors, and nonlinearities. The calibration model for a

three-axis gyroscope or accelerometer often includes a set of three biases and a 3×3 scale factor/nonorthogonality matrix containing three scale factors and 6 nonorthogonality errors, for a total of 12 calibration parameters. The bias would imply a constant offset in the measurements, but we know that this offset actually drifts in time. As discussed in [Chapter 6](#) — Sensors, this bias instability is commonly modeled using random walk processes. Mathematically, for the gyroscope case, this would result in:

$$\ddot{\omega}(t) = (\mathbf{I}_3 + \mathbf{S})[\dot{\omega}(t) + \mathbf{b}_{\omega} + \boldsymbol{\eta}_{\omega}], \text{ with } \dot{\mathbf{b}}_{\omega} = \boldsymbol{\eta}_b,$$

where \mathbf{S} is an unknown fully populated matrix of scale factors (i.e., the diagonal elements) and nonorthogonality errors (i.e., the off-diagonal elements), \mathbf{b}_{ω} is the bias, and $\boldsymbol{\eta}_b$ the random process noise describing the bias instability. An orthogonal misalignment matrix could be included in the model to account for the mounting misalignment, which is, however, typically calibrated on-ground. Note that the previous equation includes the noise term, $\boldsymbol{\eta}_{\omega}$, condensing the random stochastic errors in the measurements, which are, however, not estimated in the calibration process. In most cases, the bias fluctuation occurs slowly over many orbits. The other calibration parameters can vary as well, but typically not as fast as the bias. Hence, if a batch estimation process is used, the on-board calibration should be performed every few hours.

Magnetometer calibration is often accomplished using batch methods [[60](#),[61](#)], even if real-time sequential methods have been proposed and developed [[62](#)]. Similar to inertial sensors, magnetometers are also calibrated for biases, scale factor, and nonorthogonality errors. A common mathematical calibration model is:

$$\ddot{\mathbf{B}}(t) = (\mathbf{I}_3 + \mathbf{S})[\dot{\mathbf{B}}(t) + \mathbf{b}_B + \boldsymbol{\eta}_B],$$

where the error terms are analogous to the previous gyroscope case. Also in this case, an orthogonal misalignment matrix could be included in the model if mounting errors are not ground calibrated. The magnetometer calibration problem would be easily solvable with an accurate a priori attitude estimate, even if this is generally not the case. Fortunately, the norms of the body magnetic measurements and geomagnetic-reference vectors provide an attitude-independent scalar observation to calibrate the magnetometer.

Let's now discuss a practical applicative example of a possible gyroscope and magnetometer calibration to be performed on-board. Exploiting the previous sensor measurement models, the measured quantities can be

calibrated with two dedicated sequential iterative real-time algorithms. Both of them are suitable for a computationally efficient on-board implementation, which can also be used in small satellite missions. The gyroscope is only calibrated for the bias, accounting for its instability, thanks to the angular velocity bias estimation performed with a complementary filter [63]. The magnetometer measures are unbiased and corrected for scale factors and nonorthogonality errors, thanks to a sequential centered iterative algorithm [62]. Specifically, the calibrated quantities become:

$$\mathbf{\omega}_C(t) = \check{\mathbf{\omega}}(t) - \tilde{\mathbf{b}}_{\omega},$$

$$\mathbf{B}_C(t) = \left(\mathbf{I}_3 + \tilde{\mathbf{S}} \right)^{-1} \check{\mathbf{B}}(t) - \tilde{\mathbf{b}}_B,$$

where $\tilde{\mathbf{b}}_{\omega}$, $\tilde{\mathbf{S}}$, and $\tilde{\mathbf{b}}_B$ are the estimated error terms. Note that, to further improve the computational efficiency, the measurement model can be directly defined with the inverse of the matrix $(\mathbf{I}_3 + \mathbf{S})$, in a way to not perform the inverse operation in the calibration step. Indeed, in this case, the calibration algorithms directly estimate the elements of the inverse matrix, and the calibration step becomes:

$$\mathbf{B}_C(t) = \left(\mathbf{I}_3 + \tilde{\mathbf{S}} \right) \check{\mathbf{B}}(t) - \tilde{\mathbf{b}}_B.$$

This numerical trick is also valid for the gyroscope model with scale factors and nonorthogonality errors, even if it is of minor importance and optional. Indeed, both the measurement models can be found in existing literature methods. Further details about these calibration methods can be found in the cited literature references.

The autonomous on-board sensor calibration is also particularly useful when the FDIR function detects a fault in the system and imposes a recovery action by commanding a system reconfiguration and recalibration. In this case, the previous calibration data shall be discarded and reinitialized to calibrate the new redundant sensor components. In fact, whenever the system is reconfigured, the last calibration data involved in the reconfiguration shall be substituted. This operation is performed by the GNC system manager, and there is a settling time before design performances are reestablished. Thus, the mission manager shall consider the needed transient time in order to properly operate the system with full performance. Fig. 14.31 shows an example autonomous on-board sensor calibration with a recalibration event.

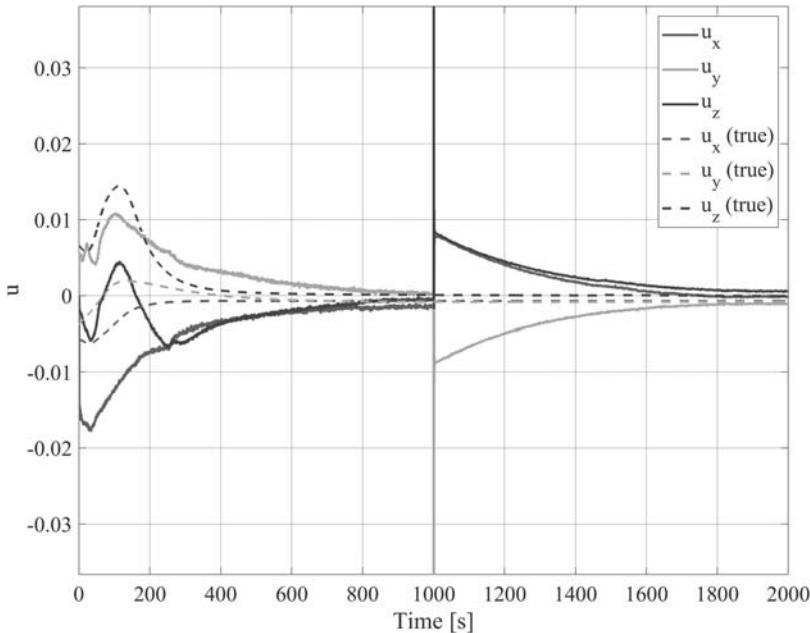


Figure 14.31 On-board calibration with a system reconfiguration and autonomous recalibration [57].

It shall be noticed how the switch to a redundant sensor with large measurement offset bias, at $t = 1000$ s, is compensated with a system reconfiguration and autonomous recalibration completed in less than 15 min. Similarly, the initial on-board calibration from unknown bias values at $t = 0$ s is completed in approximately the same amount of time [57].

GNSS-INS integration for on-board orbit determination

Orbit determination can be performed on-board, thanks to many different techniques and sensors, as discussed in [Chapter 6 – Sensors](#) and [Chapter 9 – Navigation](#). Earth-orbiting spacecraft commonly take advantage of GNSS sensors to perform orbital state measurements. These measurements can also be exploited in real-time, combined with an INS based on a set of accelerometers contained inside an inertial measurement unit (IMU, cfr. [Chapter 6 – Sensors](#)), to provide improved orbital determination performance. In particular, the GNSS-INS orbit determination system provides the long-term stability typical of GNSS systems, but it can also compensate for partial unobservability of GNSS satellites, propagating the spacecraft state

without having to rely on GNSS measurements only. As discussed in [Chapter 9](#) – Navigation, GNSS-INS integration can be made exploiting loosely or tightly coupled architectures.

Moreover, an improved on-board orbit determination system can further increase the INS-only propagation by exploiting a refined on-board orbital propagator. In this case, the orbit determination system can be operated in two modes that are identical in terms of outputs but are differentiated at measurement level, according to position and velocity sources. In this case, orbit determination would be intrinsically working with two alternative measurement sections [57]:

- Orbit determination with GNSS and accelerometer measurements, which requires the complete availability of all the orbit determination sensors. In this mode, the measurements of the IMU are combined in a navigation filter (e.g., Kalman filter) to improve position and velocity estimates and to estimate the accelerometer (e.g., bias) and gravitational model errors.
- Orbit determination with orbital propagator and accelerometer measurements, which requires a valid estimate for position and velocity, either from the previous step of the orbit determination or from ground knowledge of the current orbital state (e.g., two-line element - TLE update). In this case, the navigation filter should be modified in terms of measurement models and covariance properties with respect to the previous mode. Moreover, the estimation of accelerometers errors would be not updated in this mode, and the orbital propagator outputs are combined with statically calibrated IMU measurements to continue the orbit determination process.

The navigation filter states shall include all the sensor errors to be estimated to calibrate the available measurements, as discussed in the section about autonomous on-board sensor calibration. For example, assuming to calibrate only the accelerometer for the bias error, the calibrated acceleration vector would be:

$$\mathbf{a}_C(t) = \ddot{\mathbf{a}}(t) - \tilde{\mathbf{b}}_a.$$

Let's consider an applicative GNC example of an on-board orbit determination system based on an EKF, which has been introduced in [Chapter 9](#) – Navigation. The example orbit determination system exploits a GNSS-INS integration in a loosely coupled architecture: accelerometer readings are used to propagate the inertial orbital state vector generating the INS preliminary estimate, while GNSS position and velocity solutions

are compared with the INS estimate to generate a measurement error signal. The measurement error signal is fed to the EKF, which outputs an updated estimate of the position and velocity error, as well as the bias term estimate. Eventually, the output of the EKF is used to correct the INS preliminary estimate, whereas the bias term is fed back to the integration block of the accelerometer measurements. In the case the GNSS is experiencing an outage period or a fault, the on-board orbit propagator is used as synthetic measurement element for the INS input. The orbit propagator includes atmospheric drag and J2 perturbation effects, and it performs dead reckoning of the orbital state as soon the GNSS measurements are lost. In this case, the EKF exploits a different measurement model and covariance matrix with respect to the nominal mode with GNS-INS integration. Obviously, the approximate orbital model introduces propagation errors, and an inherent position drift cannot be avoided. However, the coupling with the EKF helps in accounting for the orbital error dynamics, and it determines a slower divergence rate with respect to a pure on-board orbital propagation including the same perturbation terms. Fig. 14.32 reports an example orbit determination solution for a low Earth-orbiting satellite. Specifically, the three-axis position error between the estimated position and the real one is shown [57].

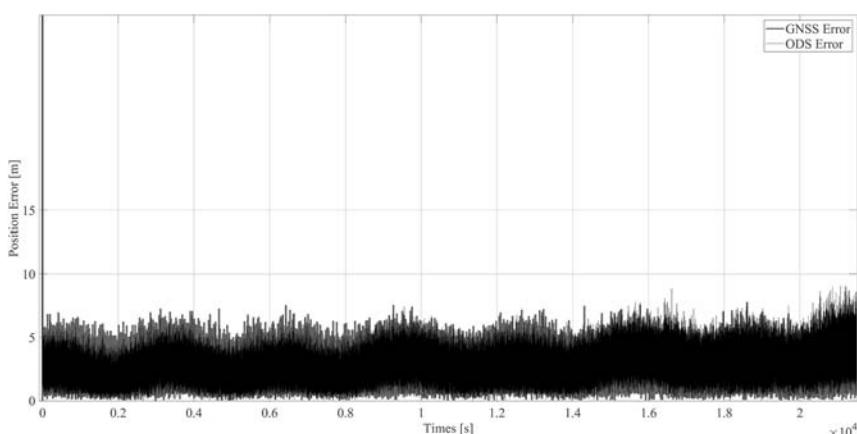


Figure 14.32 Three-axis position error for an on-board orbit determination system with a loosely coupled GNSS-INS integration [57].



Irregular solar system bodies fly around

GNC around small bodies of the Solar System represent a challenge for many aspects. First, the small mass makes the prediction of the spacecraft orbit more difficult, as perturbations like solar radiation pressure (SRP), and gravitational pull by planets or other nearby objects generate a force comparable to that of the small body itself. As the spacecraft moves nearby the object, local shape irregularities start to have a visible effect on the deviation of orbits, thus making the navigation around the body even more difficult. Also, there are control-related issues, as the very weak gravity exerted by small celestial bodies requires minimal changes in velocity to perform large relative displacements. This means that an error in the direction of the maneuver, or in its magnitude, may lead to the escape from the asteroid/comet, or a collision on its surface.

These aspects are further exacerbated if we consider that most of the knowledge about the small bodies parameters derives from on-ground observations, and the accuracy of the measurements and of the reconstructed parameters (shape, spin, etc.) is often not enough to safely perform proximity operations. Therefore, differently from missions to planetary systems, it is required an estimation not only of the state of the spacecraft but also of the asteroid/comet physical parameters. Reducing the uncertainties of the system becomes more relevant as the spacecraft is required to move closer to the target body, and the complexity and number of measurements and estimations increases as well. Therefore, navigation strategies strongly depend on the distance from the objects and on the specific mission objectives.

At sufficiently large distance, irregularities of the gravity field, caused by the uneven shape of the attractor, are negligible. Furthermore, the motion of the attractor's center of mass is reconstructed from on-ground observations in a sufficiently accurate manner. Hence, the navigation of the spacecraft can resort to techniques used for deep space, interplanetary trajectories, leveraging radiometric measurements such as range, doppler, and delta-differential one-way ranging (DDOR) [16]. Also, the set of measurements can be augmented with a centroid tracking from spacecraft camera's images.

At reduced altitudes from the attractor's surface, the gravity field irregularities become the major natural source of perturbations and require an accurate characterization through the reconstruction of the attractor's shape and physical properties. On-ground observations of small objects often do not consent a sufficiently accurate reconstruction of such properties, and

in situ measurements must be performed. The small body's parameters and the evolution of the spacecraft state are intimately related quantities; therefore, the orbit determination filter must embed all of them simultaneously. In particular, far-range orbits can be exploited to reconstruct the parameters of the attractor more accurately and reduce uncertainties, thanks to the milder influence on the spacecraft trajectory.

The reconstruction of such parameters is made possible through dedicated measurements from the spacecraft, leveraging cameras, or ranging instruments (such as Light Detection and Ranging [LIDAR] or laser range finder).

Through cameras, it is possible to perform stereo-photoclinometry [17,18], a technique that, though several images and the identification on landmarks (craters or other features) can reconstruct the shape of the body, as well as its albedo. Another noteworthy image-based technique is the silhouette and shadow carving, successfully executed for Rosetta mission [19,20], where a control volume is gradually carved where images don't show the presence of the object (silhouette carving) or where a shadow indicates a change in the surface slope (shadow carving). It is worth mentioning that this technique is generally less accurate, and stereo-photoclinometry may be performed afterward to refine the shape of the object.

Ranging instruments provide local measurements of the distance from the object surface. By collecting several measurements, it is possible to reconstruct the shape of the attractor (provided a good knowledge of the spacecraft trajectory).

The same measurements (ranging and landmarks tracking) are then exploited for orbit determination, given the higher accuracy of the attractor's parameters, at closer distances. This procedure can be performed sequentially to further improve the characterization of the small body and reduce the navigation errors of the spacecraft.

The described approach has been used for past mission toward small objects of the Solar System. It is the case of the NEAR Shoemaker probe, directed to the asteroid Eros, which performed a sequential reduction of orbital altitude to gradually improve the asteroid's parameters accuracy, leveraging radiometric measurements, laser ranging, and landmarks tracking [21]. In particular, a global initial mapping was performed at a distance about 200 km for preliminary refinement of the parameters and the definition of the landmarks set, leveraging the DSN radiometric tracking for orbit determination. Then, operative orbits at 50 km (polar) and at 35 km (retrograde)

were exploited to fulfill mission objectives, while providing further images and range measurements to refine the shape and spin of the target until the final landing on its surface. The orbit determination filter comprised a full set of 359 parameters to estimate, including spacecraft state, Eros's gravitational coefficients, spin axis and magnitude, landmarks position, etc., leading to position errors of the order of tens of meters at best.

The experience of the NEAR Shoemaker probe highlighted the necessity of an autonomous navigation of the spacecraft for the final descent phase on a small body, given the too long time needed for communication with ground, up-to-date data upload back to the spacecraft, which may take several hours [22]. First autonomous GNC attempts have been introduced in the Hayabusa 1 and Hayabusa 2 missions, respectively, directed toward the asteroids Itokawa and Ryugu [23,24]. The two spacecraft leveraged a Target Marker Tracking technique, making use of an artificial landmark dropped onto the asteroid's surface during the descent phase. The marker is designed to serve as an artificial bright spot, to be tracked through image acquisition and allow the reconstruction of the relative motion also in adverse illumination conditions. Through the marker, the spacecraft could nullify their relative motion with respect to the surface, to enable the sample collection at touchdown.

Another recent example of autonomous navigation near an asteroid is represented by the OSIRIS-Rex mission, targeting the asteroid Bennu [25]. Like Hayabusa missions, the spacecraft performed a descent and touchdown, with collection of samples from the asteroid surface. Again, autonomous navigation is exploited; however, the autonomous phase was here longer, from around 4 h prior to the touchdown. Furthermore, the spacecraft did not rely on artificial objects tracking as Hayabusa: a LIDAR was mounted as primary measurement for the autonomous navigation. The original plan was to use the LIDAR as the primary instrument for the descent and touchdown, and a Natural Feature Tracking (NFT) algorithm was developed as a backup solution (to be used with Navigation Camera images) [26]. However, after arrival at the asteroid, it was found that in order to achieve a safe sample collection, the spacecraft accuracy at touchdown needed to be on the order of a few meters, and the LIDAR-based approach could not achieve that level of accuracy. Thus, the NFT method was selected to be the primary and was successfully used to gather the sample.

Fig. 14.33 shows some example fly around trajectories around irregular Solar System minor bodies [64,65].

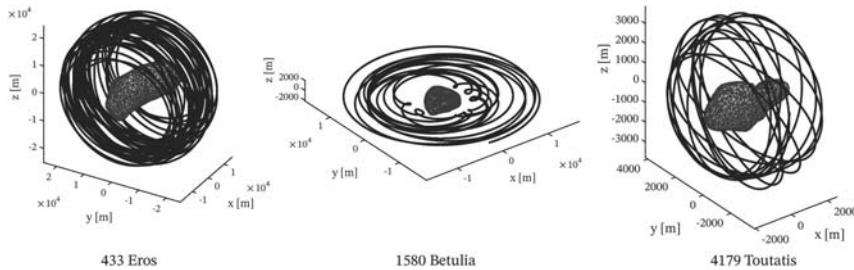


Figure 14.33 Example irregular Solar System bodies fly around trajectories.

GNC for planetary landing

Planetary landing is, without any doubt, one of the most critical tasks that to be performed during a space mission. The landing sequence is usually composed by an orbiting phase followed by the Entry, Descent, and Landing (EDL) phase. The EDL is usually considered a mission critical phase: a failure at this stage would imply, with high probability, the complete loss of the spacecraft. Despite of the relatively large number of successful missions since the Apollo program, landing is complex and difficult, as denoted by the recent failure of the European Space Agency lander module Schiaparelli on Mars [27].

The landing sequence is characterized by multiple flight phases and high uncertainties. The last phase of a landing sequence is usually called powered descent (PD), and, at this stage, the GNC system must be able to deal with accumulated errors until touchdown. An example of a classical landing profile is depicted in Fig. 14.34. The sequence begins with a deorbit burn, followed by a coast phase. The coast phase ends near the periapsis when the powered descent phase starts and terminates with touchdown.

In this section, the main GNC algorithmic alternatives and implementation challenges are summarized with a focus to powered descent phase.

Planetary landing guidance

Formulation

The powered descent phase typically starts at a few hundreds of kilometers relative to surface. For this reason, it can be assumed that the lander is subject to uniform gravity. Furthermore, aerodynamic forces are negligible compared to the retrorocket. In fact, the eventual presence of atmosphere could be negligible due to the relative low velocity (~ 100 m/s) and shape of the spacecraft. Consequently, the three-dimensional translational

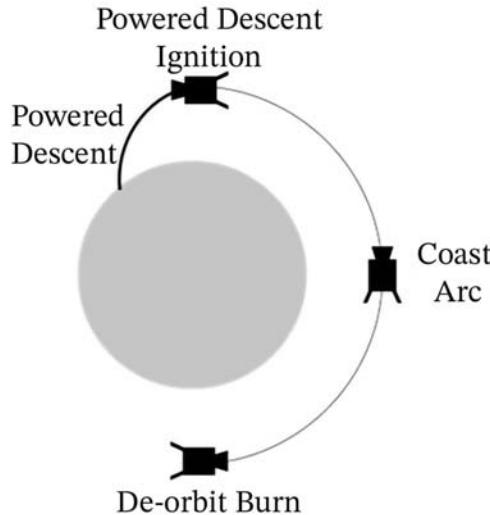


Figure 14.34 Example of landing profile.

dynamics expressed in a Ground Reference System can be described by a set of equations:

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v} \\ \dot{\mathbf{v}} = \frac{\mathbf{T}}{m} + \mathbf{g} \\ \dot{m} = -\frac{\|\mathbf{T}\|}{I_{sp}g_0} \end{cases}$$

where \mathbf{r} and \mathbf{v} are the position and velocity vectors, \mathbf{T} is the engine thrust, \mathbf{g} is the constant acceleration of gravity vector of the planet, I_{sp} the specific impulse of the engine, and g_0 the standard gravity acceleration at sea level. In order to set an optimization problem, the following initial and final states can be set:

$$\begin{cases} \mathbf{r}(t_0) = \mathbf{r}_0 \\ \mathbf{v}(t_0) = \mathbf{v}_0 \\ m(t_0) = m_0 \\ \mathbf{r}(t_f) = \mathbf{r}_f \\ \mathbf{v}(t_f) = \mathbf{v}_f \end{cases}$$

In general, the local terrain characteristics should be taken into account to avoid potential hazards on the planet surface. In a realistic mission, sensors can be utilized to obtain the digital terrain model and to discriminate between safe and unsafe landing sites and possible hazards along the trajectory (i.e., craters and hills). There are two main alternatives to deal with terrain constraints: constraint embedded optimization strategy and the waypoint-based scheme. The first strategy formulation includes the hazardous terrain as constraint in the optimization problem, and, in this way, hazards are autonomously avoided. In the waypoint-based method, waypoints along the descent trajectory are defined. In this way, these selected waypoints force the flight path to avoid hazards. Waypoints can be determined in advance on the reference trajectory. The introduction of waypoints to reach along the trajectory implies a smaller difference between the guidance trajectory and reference solution at the cost of frequent maneuvers and therefore a higher fuel consumption. Furthermore, the thrust limitations given by the engine characteristics have to be included as:

$$T_1 \leq \|\mathbf{T}\| \leq T_2$$

with T_1 and T_2 being the minimum and maximum available thrust.

Minimum fuel consumption is usually the adopted performance index. Several ways exist to express this index as cost function to be minimized. Some of them are:

$$\begin{aligned} J &= -m(t_f) \\ J &= \int_0^{t_f} \|\mathbf{T}\| dt \\ J &= \frac{1}{2} \int_0^{t_f} \mathbf{a}^T \mathbf{a} dt \\ J &= t_f \end{aligned}$$

with $\mathbf{a} = \mathbf{T}/m$ and t_f the final time, parameter to be determined by the guidance law. Having introduced dynamical equations, cost function, and typical constraints, the unified optimal powered descent guidance or trajectory planning problem is given in this section.

Guidance algorithms

Some of the main analytical and numerical guidance algorithms usually used to solve this optimization problem are introduced [28].

Polynomial method

The polynomial method is an analytical model adopted also by the Apollo lander [29]. The basic idea is to simplify the thrust acceleration profile and describing it as a polynomial of flight time. A quadratic thrust acceleration profile is assumed as:

$$\mathbf{a}(\tau) = \mathbf{c}_0 + \mathbf{c}_1\tau + \mathbf{c}_2\tau^2 \quad \tau \in [t, t_f].$$

By considering the terminal position, and the velocity and acceleration constraints, analytical expressions of the acceleration in line with the current and terminal states can be derived:

$$\mathbf{a}(t) = \frac{12}{t_{go}^2} (\mathbf{r}_f - \mathbf{r} - \mathbf{v}t_{go}) - \frac{6}{t_{go}} (\mathbf{v}_f - \mathbf{v}) + \mathbf{a}_f \quad (14.3)$$

where $t_{go} = t_f - t$ is defined as the time-to-go. Please see Refs. [28,30] for the complete derivation. As implied in Eq. (14.1), the current thrust acceleration can be determined by current and final states. Thrust bounds are not taken explicitly into account with this kind of formulation, but a saturation function can be introduced to meet the thruster constraints.

Potential field method

The potential field method combines APF method and Lyapunov stability theory to perform both pin-point landing while avoiding possible hazards. In particular, a Lyapunov function can be constructed considering a contribution of a state function V_s and a potential function V_p as in Eqs. (14.2)–(14.4).

$$V = V_s + V_p \quad (14.4)$$

$$V_s = \left[\mathbf{r}^T - \mathbf{r}_f^T, \mathbf{v}^T - \mathbf{v}_f^T \right]^T \mathbf{P} \left[\mathbf{r}^T - \mathbf{r}_f^T, \mathbf{v}^T - \mathbf{v}_f^T \right] \quad (14.5)$$

$$V_p = \sum_{i=1}^n \mathbf{e} |\mathbf{r}_{ci}| \exp \left[-\frac{(\mathbf{r}^T - \mathbf{r}_{ci}^T) \mathbf{H} (\mathbf{r} - \mathbf{r}_{ci})}{\sigma_i^2} \right] \quad (14.6)$$

where \mathbf{P} and \mathbf{e} are parameters to be tuned to shape the descent trajectory, \mathbf{r}_{ci} is the position of local obstacles and hazards to be avoided, \mathbf{H} is a constant

matrix to calculate the square of the horizontal distance between the lander and the hazard center, and σ is a positive shape parameter.

Zero-effort-miss/zero-effort-velocity method

The zero-effort-miss/zero-effort-velocity (ZEM/ZEV) method is a feedback guidance law. It solves the guidance command with the energy optimal cost function $J = \frac{1}{2} \int_0^{t_f} \mathbf{a}^T \mathbf{a} dt$ to meet the position and velocity constraints. ZEM and ZEV represent the position and velocity deviations between the lander and the landing site if no control force is exerted:

$$\begin{aligned} \text{ZEV}(t) &= \mathbf{v}_f - \left[\mathbf{v}(t) + \int_t^{t_f} \mathbf{g}(\tau) d\tau \right] \\ \text{ZEM}(t) &= \mathbf{r}_f - \left[\mathbf{r}(t) + t_{go} \mathbf{v} + \int_t^{t_f} (t_f - \tau) d\tau \right] \end{aligned}$$

The expression for ZEM and ZEV, combined with the cost function, allows to derive an analytical expression of optimal acceleration \mathbf{a}_g :

$$\mathbf{a}_g = \frac{6}{t_{go}^2} \text{ZEM} - \frac{2}{t_{go}} \text{ZEV}$$

Please note that the solution of this equation in the optimal control framework is quite complicated because it implies the solution of a quadratic equation with respect to t_{go} . A possible alternative approach is to add a constraint on the altitude component of the thrust acceleration vector and to solve for t_{go} . Similar to the polynomial method, the saturation function is introduced to take into account the thrust limit.

Pseudospectral method

The pseudospectral method (PSM) is a generic approach to solve the optimal control problem when dealing with NLP problem. The original pinpoint landing problem can be translated into an NLP by approximating both state and control variables simultaneously at a series of collocations using interpolating polynomials. This allows to rewrite the cost function, dynamics, and constraints into algebraic equalities or inequalities. These discretized expressions define an NLP problem that can be solved by PSM. The solution is an approximate solution to the original continuous optimal control problem. The detailed formulation is described in Ref. [28].

Convex optimization method

Problems with convex objective functions, linear equality constraints, and convex inequality constraints can be solved by means of convex optimization techniques. This kind of problem is very advantageous because it does not require an initial guess of the optimization variables and it provides fast convergence and polynomial time complexity. The original landing problem can be rewritten in a convex form also making use of lossless convexification techniques and expressing the problem constraints in a convex form. This kind of guidance method was proposed also for Mars landing [31].

Sensors and navigation

During a planetary landing, it is of extreme importance to accurately know the lander position with respect to the planet's surface. This task is performed by exploiting different sensors and a navigation filter [32]. Also in this scenario, two types of sensors are available: passive imaging or active range sensing. The pros and cons of these two alternatives are similar to the ones detailed in [Chapter 9](#) – Navigation while describing relative navigation alternatives. Passive sensors such as visible cameras are a mature technology that has the advantage of low power consumption, limited mass and volume. Depending on the camera characteristics, navigation measurements with different level of accuracy can be provided from any altitude during a landing sequence. The main drawback of passive sensors is that they require, in general, good illumination conditions and cannot operate in the dark. On the contrary, active range sensing like LIDAR or altimeters can operate under any illumination conditions. The main drawback of this relatively green technology for space applications is that they have a limited operating range and, in general, high-power consumption. This represents a constraint on the altitude at which navigation measurements can be made available. From the navigation filter point of view, the main difference is represented by the kind of information provided by the approach: global or local position estimation. Global position estimation provides the best estimate of the absolute position of the lander with respect to a global coordinate system attached to a planet surface map available before landing. In this way, it is possible to maintain a controlled position with respect to a desired landing site, specified on the surface map. In contrast, local position estimation provides the best position estimate with respect to a map of the landing site made onboard, during landing. This local map is usually more accurate, but it cannot, in general, be correlated to the global map and, therefore,

the absolute position cannot be improved by using local approaches. Comparing subsequent local position, relative velocity with respect to the terrain can be computed. Finally, a further classification can be made depending on the structure of the algorithm, distinguishing between *correlation* and *pattern-matching* approaches. For passive imagers, the correlation step is performed by correlating the acquired image with an onboard map (that can be the previous step). This is done measuring the similarity between the current image and the map value. For active range sensors, the approach is similar, but, instead of images, elevation maps or contours are correlated. Correlation is a common 2D signal processing operation and, therefore, it is possible to implement it on field-programmable gate array to allow for fast computation. However, they have the disadvantage that the surface data must be rectified. The rectification is a process that removes the scale and perspective differences between two sets of data.

This step causes a direct coupling between errors in the position estimate derived from correlation and the state estimate used to reorient the surface data (correlated measurements and states used in a filter may affect its stability). A different approach is represented by pattern-matching methods. Instead of correlating patch of surface, this kind of algorithms tries to match landmarks between map and surface data. Landmarks are locations that can be extracted reliably such as craters. Therefore, while for correlation, the map and surface data are directly compared, in pattern matching, landmarks are first extracted from the surface data and map, respectively, and then compared. Given this high-level classification, some of the most common approaches can be summarized as in Ref. [32].

- *Crater pattern matching* [33]. This pattern-matching algorithm provides an absolute position estimation by using surface craters as landmarks. These landmarks are extracted in the map and collected into a database. During landing, crater features are detected and compared to the landmarks present in the database. When a match is obtained, the absolute lander position can be estimated. One major advantage of using craters as landmarks, compared to other approaches where unknown features are used, is that craters can be recognized by their intrinsic characteristics which are highly independent of the illumination conditions.
- *Scale-invariant feature transform (SIFT) descriptor matching* [34]. SIFT descriptors [35] are commonly used in image processing techniques and can be extracted from imagery that do not have craters. The process is similar to the crater pattern-matching technique, but a different landmark definition is used (SIFT descriptors vs. craters). A main drawback is that SIFT

descriptors are sensible to illumination and viewing angle. For this reason, important constraints in the generation of the database as the need to recreate images comparable to those to be captured by the on-board camera. The simulation SW needs not only to be quite realistic but also, more importantly, the surface data have to be available with an extreme level of detail.

- *Image to map correlation* [36]. This method uses correlation to compare descent images directly to a previously obtained image of the landing site (e.g., taken from an orbiter). The image has to be rectified first to match the scale and orientation of the previously obtained image and then the correlation step is performed.
- *Descent image motion estimation* [37]. In this method, during the descent phase, subsequent images are correlated between each other in order to estimate the velocity of the lander relative to the surface. The rectification process is not necessary if the images are acquired fast enough.
- *Structure from motion* [38]. The concept of this technique is very similar to the Descent Image Motion Estimation. The main difference is that not only velocity but also position and attitude of the lander are estimated. This is possible but only up to an unknown scale factor that has to be resolved in a different way (e.g., with an altimeter).
- *Shape signature pattern matching* [39]. While dealing with active ranging sensors, shape signatures can be used as landmarks. In fact, they are surface landmarks based on local surface shape that can be used for a pattern-matching approach. Shape signatures are extracted from a previously available digital elevation map (DEM) and correlated with surface data (e.g., range image available from a LIDAR).
- *Range image to DEM correlation* [40]. In this case, a DEM is created from a single or multiple range images and then rotated into the map frame before correlation. The two available DEMs are then correlated to estimate the position.
- *Altimeter to DEM correlation* [41]. The approach is very similar to the previous one. The main difference is that instead of a DEM to DEM, an elevation data to DEM correlation is performed. Elevation data are generated by exploiting only the data coming from the altimeter.
- *Consecutive range image correlation* [42]. Similar to image processing techniques, two consecutive range images are rotated into the map frame and correlated between each other. This correlation gives an indication in the change of position of the lander.

Hazard avoidance

All the unmanned landing sequences involve a hazard avoidance method. In particular, autonomous algorithms need to be employed in order to discriminate between a safe and a hazardous patch of the terrain. This process is based on the evaluation of specific characteristics of the planet surface such as: slope of the terrain, surface roughness, and safe landing area size. Hazard maps can be generated according to various criteria that are mission-dependent and based on available sensors. General methods for vision-based hazard map generation are based on intensity-based algorithms [43] that determine the surface roughness based on pixel standard deviation. Also LIDARs can be used to generate hazard maps. This is done by exploiting the DEM available after the processing of multiple range images. The derived DEM is used to extract slope and roughness characteristics of the terrain using a least-squares plane fitting algorithm [44]. Finally, the hazard map can also be the product of multiple sensor processing [45]. In fact, a fuzzy logic methodology for fusing sensed data can be adopted to generate a multisensors hazard map. Once a hazard map is generated, a new landing site can be selected according to different criteria. A common strategy is to generate a cost function that selects the landing site that has minimal landing incidence angle and roughness landing while keeping the lander away from detected hazards.

References

- [1] B. Hamilton, Sizing and steering CMG arrays for agile spacecraft, ESA GNC 2017 Conference.
- [2] D. Alazard, et al., Two-input two-output port model for mechanical systems, in: AIAA Guidance, Navigation, and Control Conference, 2015.
- [3] D. Alazard, F. Sanfedino, A short course on TITOP models for space system modelling, IFAC-PapersOnLine 54 (12) (2021) 7–13.
- [4] F. Sanfedino, D. Alazard, Pommier-Budinger, Valérie, A. Falcoz, F. Boquet, Finite element based N-Port model for preliminary design of multibody systems, Journal of Sound and Vibration 415 (2018) 128–146. ISSN 0022-460X.
- [5] D. Alazard, F. Sanfedino, Satellite dynamics toolbox for preliminary design phase, in: 43rd Annual AAS Guidance and Control Conference, 30 January 2020 – 5 February 2020 (Breckenridge, United States), 2020.
- [6] D. Alazard, F. Sanfedino, Satellite Dynamics Toolbox Library (SDTlib) – User’s Guide,” Tech. Rep, Institut Supérieur de l’Aéronautique et de l’Espace, 2021. <https://nextcloud.isae.fr/index.php/s/oPQjcytZMxL27a5> (Accessed April 6 2021).
- [7] N. Guy, D. Alazard, C. Cumér, C. Charbonnel, Dynamic modeling and analysis of spacecraft with variable tilt of flexible appendages, Journal of Dynamic Systems, Measurement, and Control 136 (2) (2014) 021020.
- [8] G. Balas, R. Chiang, A. Packard, M. Safonov, Robust Control Toolbox User’s Guide, The Math Works, Inc., Tech. Rep, 2007.

- [9] P. Apkarian, P. Gahinet, C. Buhr, Multi-model, multi-objective tuning of fixed-structure controllers, in: 2014 European Control Conference (ECC). IEEE, 2014, pp. 856–861.
- [10] P. Apkarian, M.N. Dao, D. Noll, Parametric robust structured control design, *IEEE Transactions on Automatic Control* 60 (7) (2015) 1857–1869.
- [11] S. Silvestrini, M. Lavagna, Neural-aided GNC reconfiguration algorithm for distributed space system: development and PIL test, *Advances in Space Research* 67 (5) (2021) 1490–1505.
- [12] L.M. Steindorf, S. D’Amico, J. Scharnagl, F. Kempf, K. Schilling, Constrained low-thrust satellite formation-flying using relative orbit elements, in: 27th AAS/AIAA Space Flight Mechanics Meeting, 160, February 2017, pp. 3563–3583.
- [13] C.M. Saaj, V. Lappas, V. Gazi, Spacecraft swarm navigation and control using artificial potential field and sliding mode control, in: 2006 IEEE International Conference on Industrial Technology, IEEE, December 2006, pp. 2646–2651.
- [14] D. Morgan, S.J. Chung, F.Y. Hadaegh, Model predictive control of swarms of spacecraft using sequential convex programming, *Journal of Guidance, Control, and Dynamics* 37 (6) (2014) 1725–1740, <https://doi.org/10.2514/1.G000218>.
- [15] S. Silvestrini, M. Lavagna, Neural-based predictive control for safe autonomous space-craft relative maneuvers, *Journal of Guidance, Control, and Dynamics* 44 (12) (2021) 2303–2310.
- [16] D.W. Cerkendall, J.S. Border, Delta-DOR: the one-nanoradian navigation measurement system of the deep space network—history, architecture, and componentry, *The Interplanetary Network Progress Report* 42 (2013) 193.
- [17] R.W. Gaskell, O.S. Barnouin, D.J. Scheeres, et al., Characterizing and navigating small bodies with imaging data, *Meteoritics and Planetary Science* 43 (6) (2008) 1049–1061.
- [18] O.S. Barnouin, R. Gaskell, E. Kahn, et al., Assessing the Quality of Topography from Stereo-Photoclinometry, *Asteroids Comets Meteors*, 2014.
- [19] R. Pardo de Santayana, M. Lauer, Optical measurements for rosetta navigation near the comet, *International Symposium on Space Flight Dynamics* (2015).
- [20] M. Lauer, S. Kielbassa, R. Pardo, Optical Measurements for Attitude Control and Shape Reconstruction at the Rosetta Flyby of Asteroid Lutetia, *International Symposium on Space Flight Dynamics*, 2012.
- [21] B.G. Williams, Technical challenges and results for navigation of NEAR Shoemaker, *Johns Hopkins APL Technical Digest* 23 (1) (2002) 34–45.
- [22] S. Bhaskaran, S. Nandi, S. Broschart, et al., Small body landings using autonomous on-board optical navigation, *Journal of the Astronautical Sciences* 58 (3) (2011) 409–427.
- [23] M. Uo, K. Shirakawa, T. Hasimoto, Hayabusa touching-down to Itokawa—autonomous guidance and navigation, *The Journal of Space Technology and Science* 22 (1) (2006) 32–41.
- [24] N. Ogawa, F. Terui, Y. Fuyuto, et al., Image-based autonomous navigation of Hayabusa2 using artificial landmarks: the design and brief in-flight results of the first landing on asteroid Ryugu, *Astrodynamicics* 4 (2) (2020) 89–103.
- [25] B. Williams, P. Antreasian, E. Carranza, et al., OSIRIS-REx flight dynamics and navigation design, *Space Science Reviews* 214 (4) (2018) 1–43.
- [26] D.A. Lorenz, R. Olds, A. May, et al., Lessons learned from OSIRIS-Rex autonomous navigation using natural feature tracking, *IEEE Aerospace Conference* (2017) 1–12.
- [27] T. Tolker-Nielsen, EXOMARS 2016-Schiaparelli Anomaly Inquiry, 2017.
- [28] X. Liu, S. Li, M. Xin, Comparison of powered descent guidance laws for planetary pin-point landing, *Acta Astronautica* (2021).
- [29] A.R. Klumpp, Apollo lunar descent guidance, *Automatica* 10 (2) (1974) 133–146.

- [30] Z.-yu Song, et al., Survey of autonomous guidance methods for powered planetary landing, *Frontiers of Information Technology & Electronic Engineering* 21 (5) (2020) 652–674.
- [31] 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.
- [32] A.E. Johnson, J.F. Montgomery, Overview of terrain relative navigation approaches for precise lunar landing, in: 2008 IEEE Aerospace Conference. IEEE, 2008.
- [33] Y. Cheng, A. Ansar, Landmark based position estimation for pinpoint landing on mars, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation. IEEE, 2005.
- [34] N. Trawny, et al., Vision-aided inertial navigation for pin-point landing using observations of mapped landmarks, *Journal of Field Robotics* 24 (5) (2007) 357–378.
- [35] D.G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the Seventh IEEE International Conference on Computer Vision, 2, IEEE, 1999.
- [36] J.R. Carr, J.S. Sobek, Digital scene matching area correlator (DSMAC), *Image Processing For Missile Guidance* 238 (1980). International Society for Optics and Photonics.
- [37] A. Johnson, et al., Design through operation of an image-based velocity estimation system for Mars landing, *International Journal of Computer Vision* 74 (3) (2007) 319–341.
- [38] H.C. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, *Nature* 293 (5828) (1981) 133–135.
- [39] A. Frome, et al., Recognizing objects in range data using regional point descriptors, in: European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2004.
- [40] B. Sabata, J.K. Aggarwal, Estimation of motion from a pair of range images: a review, *CVGIP: Image Understanding* 54 (3) (1991) 309–324.
- [41] J.P. Golden, Terrain contour matching (TERCOM): a cruise missile guidance aid, *Image processing for missile guidance* 238 (1980). International Society for Optics and Photonics.
- [42] A.E. Johnson, A. Miguel San Martin, Motion estimation from laser ranging for autonomous comet landing, in: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), 1, IEEE, 2000.
- [43] Y. Cheng, et al., Passive Imaging Based Hazard Avoidance for Spacecraft Safe Landing, 2001.
- [44] A.E. Johnson, et al., Lidar-based hazard avoidance for safe landing on Mars, *Journal of Guidance, Control, and Dynamics* 25 (6) (2002) 1091–1099.
- [45] A. Howard, H. Seraji, Multi-sensor terrain classification for safe spacecraft landing, *IEEE Transactions on Aerospace and Electronic Systems* 40 (4) (2004) 1122–1131.
- [46] J.F. Trégouët, D. Arzelier, D. Peaucelle, et al., Reaction wheels desaturation using magnetorquers and static input allocation, *IEEE Transactions on Control Systems Technology* 23 (2) (March 2015) 525–539, <https://doi.org/10.1109/TCST.2014.2326037>.
- [47] A.C. Stickler, K.T. Alfriend, Elementary magnetic attitude control system, *Journal of Spacecraft and Rockets* 13 (5) (1976) 282–287.
- [48] F.L. Markley, J.L. Crassidis, Fundamentals of Spacecraft Attitude Determination and Control, Space Technology Library, Springer, New York, 2014.
- [49] A. Colagrossi, M. Lavagna, Fully magnetic attitude control subsystem for picosat platforms, *Advances in Space Research* 62 (2018) 3383–3397.
- [50] A. Colagrossi, M. Lavagna, A spacecraft attitude determination and control algorithm for solar arrays pointing leveraging Sun angle and angular rates measurements, *Algorithms* 15 (2022) 29, <https://doi.org/10.3390/a15020029>.

- [51] A.H.J. De Ruiter, C.J. Damaren, J.R. Forbes, *Spacecraft Dynamics and Control: An Introduction*, John Wiley & Sons, 2013.
- [52] H. Curtis, *Orbital Mechanics for Engineering Students*, Elsevier, 2005.
- [53] M.J. Sidi, *Spacecraft Dynamics and Control: A Practical Engineering Approach*, Cambridge University Press, 1997.
- [54] D.A. Vallado, *Fundamentals of Astrodynamics and Applications*, 12, Springer Science & Business Media, 2001.
- [55] J.M. Longuski, J.J. Guzmán, J.E. Prussing, *Optimal Control with Aerospace Applications*, Microcosm Press, Springer, 2014.
- [56] A. Colagrossi, V. Pesce, L. Bucci, et al., Guidance, navigation and control for 6DOF rendezvous in Cislunar multi-body environment, *Aerospace Science and Technology* 114 (2021) 106751.
- [57] A. Colagrossi, M. Lavagna, Fault tolerant attitude and orbit determination system for small satellite platforms, *Aerospace* 9 (2022) 46.
- [58] R. Pandiyan, A. Solaiappan, N. Malik, A one step batch filter for estimating gyroscope calibration parameters using star vectors, in: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*. Providence, AIAA 04-4858, 2004.
- [59] M.E. Pittelkau, Kalman filtering for spacecraft system alignment calibration, *Journal of Guidance, Control, and Dynamics* 24 (6) (2001).
- [60] R. Alonso, M.D. Shuster, Complete linear attitude-independent magnetometer calibration, *Journal of the Astronautical Sciences* 50 (4) (2002) 477–490.
- [61] R. Alonso, M.D. Shuster, TWOSTEP: a fast robust algorithm for attitude-independent magnetometer-bias determination, *Journal of the Astronautical Sciences* 50 (4) (2002) 433–451.
- [62] J.L. Crassidis, K.L. Lai, R.R. Harman, Real-time attitude-independent three-axis magnetometer calibration, *Journal of Guidance, Control, and Dynamics* 28 (1) (2005) 115–120.
- [63] R. Mahony, T. Hamel, J.M. Pflimlin, Nonlinear complementary filters on the special orthogonal group, *IEEE Transactions on Automatic Control* 53 (2008) 1203–1218.
- [64] A. Colagrossi, et al., Dynamical evolution about asteroids with high fidelity gravity field and perturbations modeling, in: *2015 AAS/AIAA Astrodynamics Specialist Conference*. Univelt, 2016.
- [65] A. Colagrossi, F. Ferrari, M. Lavagna, Coupled dynamics analysis around asteroids by means of accurate shape and perturbations modeling, in: *66th International Astronautical Congress (IAC 2015)*. Curran Associates, 2015.
- [66] A. Colagrossi, M. Lavagna, Dynamical analysis of rendezvous and docking with very large space infrastructures in non-Keplerian orbits, *CEAS Space Journal* 10 (1) (2018) 87–99, <https://doi.org/10.1007/s12567-017-0174-4>.

This page intentionally left blank



Modern Spacecraft GNC

Stefano Silvestrini¹, Lorenzo Pasqualetto Cassinis², Robert Hinz³,
David Gonzalez-Arjona⁴, Massimo Tipaldi⁵, Pierluigi Visconti⁶,
Filippo Corradino⁶, Vincenzo Pesce⁷, Andrea Colagrossi¹

¹Politecnico di Milano, Milan, Italy

²TU Delft, Delft, the Netherlands

³Deimos Space, Madrid, Spain

⁴GMV Aerospace & Defence, Madrid, Spain

⁵University of Sannio, Benevento, Italy

⁶Tyvak International, Turin, Italy

⁷Airbus D&S Advanced Studies, Toulouse, France

The following chapter gives an overview on modern techniques for guidance, navigation, and control (GNC). In particular, an overview of artificial intelligence (AI) techniques is provided in light of a tailored application to the space domain. Thanks to their enormous success in a great variety of applications and fields, modern AI techniques can be found in almost every aspect of science and engineering as well as everyday life. AI enables the automation of tasks previously limited to humans, even surpassing human performance on many tasks. Consequently, the terms AI, machine learning (ML), and deep learning (DL) are nowadays ubiquitous and are often used interchangeably to describe computer systems which are designed to act in an intelligent way. Among the modern applications, a thorough description of innovative methods for GNC failure (or fault) detection, isolation, and recovery (FDIR) is presented, highlighting the latest novelties. Finally, the emerging topic of CubeSats and nanosatellites, in general, is treated by underlining the peculiar challenges that such missions pose.

In detail, this chapter is composed by the following sections:

- *AI in space – Introduction.* This section presents the basics and fundamental concepts of AI and DL, with a particular attention to space-related applications.
- *AI and navigation.* This section extends the fundamentals of AI described in the first section to convolutional neural networks (CNNs)-based systems for space navigation.

- *Validation of AI-based systems – Introduction.* The section presents the challenges linked to the validation of AI-based systems. In particular, the problem of sparsity of real space-born imagery and the necessity of large data sets for ANN/DL-based methods is tackled. Then, the section discusses the procedures for the generation of laboratory and synthetic images and the challenges of training on synthetic datasets.
- *Reinforcement learning.* This section presents an overview of reinforcement learning paradigm for AI-based systems.
- *AI use cases.* The section presents a set of use cases of AI-based GNC systems at research level. This section should be used as an up-to-date literature survey.
- *AI on-board processors.* The section presents the avionics hardware (HW) processing solutions targeting specifically AI demanding applications such as multiprocessors, video processor units (VPUs), LPGPU, FPGA, VITIS-XILINX versus INTEL, DSP, and the link to the development frameworks that might include autocoding or deployment libraries for different processing elements modules of the execution neuronal network (convolution engines, max-pooling, etc.).
- *Innovative techniques for highly autonomous FDIR in GNC applications.* The section presents the FDIR system evolution foreseen in the next years, which includes model-based FDIR system, in terms of innovative technical solution and its workflow. Moreover, advanced control engineering methods, together with ML-based techniques for innovative GNC applications are presented.
- *Small satellites/CubeSats.* This section intends to provide a practical outlook on GNC and Attitude Determination and Control System (ADCS) subsystems in nanosatellites, especially to designers coming from academia or larger spacecraft. The section goes over a brief overview of CubeSats missions and their state-of-the-art, in order to introduce this class of platform should the reader not be familiar with it. Then, it extensively analyzes the limitations related to the HW which can be employed in CubeSats for Altitude and Orbit Control (AOCS)/ADCS, first from a higher-level perspective (volumes, redundancy, rad-hard vs. commercial, etc.). Finally, it provides an outlook on the practical challenges related to sizing and design trade-offs, for a low Earth orbit (LEO) mission and for an interplanetary mission with propulsion, as well as verification and testing, looking also at typical requirements and how to approach their compliance and verification.



AI in space—Introduction

Introduction

AI, ML, DL, and ANN: What is the difference?

The terms AI, ML, and DL are currently present in most of technological discussions in different engineering fields. Moreover, they are often mutually replaced to describe computer systems which are designed to act in an intelligent way. While these terms are closely linked, with DL and artificial neural networks (ANNs) forming part of ML, which in turn is considered a subset of AI (Fig. 15.1), it is useful to understand the key distinctions between them:

- ***AI.*** It is a branch of computer science which is concerned with the theory and development of systems which are capable of performing tasks normally requiring human intelligence. This does include systems which emulate human intelligence based on fixed rules and data bases, like knowledge graphs and expert systems, but also ML systems which are capable of modifying themselves when exposed to new data.
- ***ML.*** It is, therefore, generally considered a subset of AI. It is commonly defined as the field of study that gives computers the ability to learn without being explicitly programmed [1].¹ A more formal definition of the types of algorithms studied in the field of ML is given by Ref. [2]:

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

An ML algorithm generally learns by solving an optimization problem, typically by minimizing the error or maximizing the likelihood of its predictions being true, with the performance of the algorithm being measured by a task-dependent objective function.

- ***DL.*** It is a subfield of ML concerned with ANNs, or more precisely, deep neural networks (DNNs), which consist of ANN with more than two layers. ANNs are ML models inspired by the biological neural networks that constitute animal brains. Although conceptually around for over half a century, they only gained importance over the last decade due to their high computational demands. DNNs are setting new records in

¹ This definition paraphrases a quote from Arthur Samuel, a pioneer of artificial intelligence research: “Programming computers to learn from experience should eventually eliminate the need for much [...] programming effort.” [1].

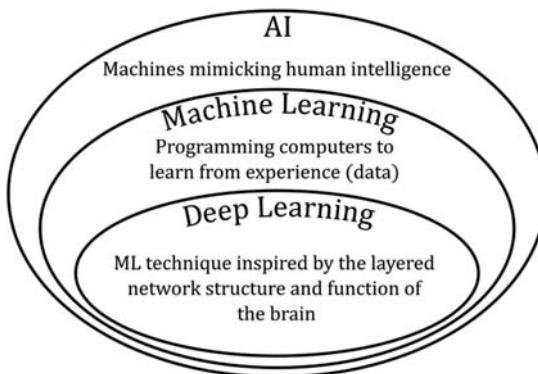


Figure 15.1 Relation between AI, ML, and DL.

accuracy for many important problems, such as image recognition, sound recognition, recommender systems, and natural language processing.

Since the term AI is extremely wide and its boundaries often blurred, this book will focus on ML and DL methods.

Learning paradigms: supervised, unsupervised, and reinforcement learning

ML scenarios can be categorized depending on the task to be solved and the data available to both train and test the learning algorithm. The three main categories are supervised, unsupervised, and reinforcement learning, which are summarized in the following alongside the most common tasks they are associated with.

- *Supervised learning.* A supervised learning algorithm receives a training data set consisting of labeled examples, where for each given input, the true output is provided. After the learning phase, the algorithm makes predictions on unseen data. This is the most common setup in regression and classification problems, as illustrated in Fig. 15.2.
- 1. *Classification.* In classification problems, the aim is to assign a discrete value or class label to each input. Typical examples include spam detection, where the algorithm determines whether an email should be considered spam or not, and image classification, where class labels are assigned to each image depending on their content (e.g., if they show a car, a plane, or a boat). Fault detection can also be framed as a classification problem, where, for example, from a number of telemetry parameters, the correct or faulty behavior of a spacecraft is determined.

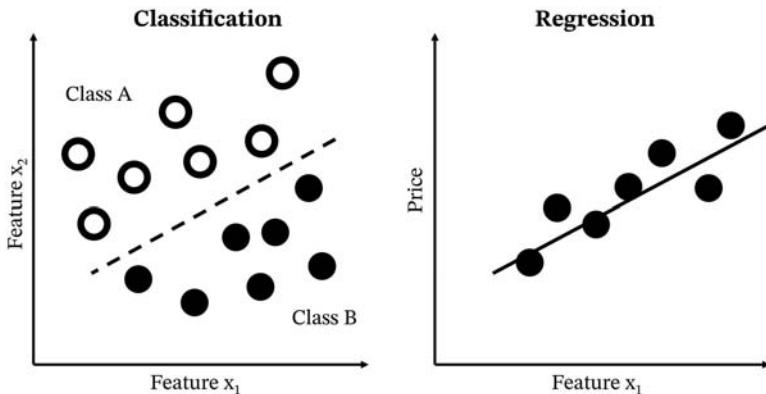


Figure 15.2 Supervised learning tasks: classification and regression.

2. *Regression.* In regression problems, the task is to predict a real valued output for each input. Typical regression examples include stock market and house price prediction, where a price is predicted on the basis of input features like number of rooms and the area of a house.

A detailed description of supervised learning is given in the next section where linear and logistic regression is introduced for regression and classification tasks.

- *Unsupervised learning.* In unsupervised learning, training is performed on unlabeled training data. This means that during training, the correct or true output for the given input is unknown. In general, this makes it difficult to quantitatively evaluate the performance. The learning algorithm is used to automatically find patterns and structure in the data. Applications include clustering, where the objective is to partition the dataset into homogeneous subsets, and dimensionality reduction methods such as principal component analysis (PCA) [3], where a lower-dimensional representation of high-dimensional input data is found (Fig. 15.3). The k-means algorithm is described below as an example for a widely used unsupervised learning method.
- *Reinforcement learning.* In reinforcement learning, an algorithm learns from trial and error. In contrast to supervised and unsupervised learning, there is no clear separation between the training and testing phases. Instead, the learning algorithm, called agent, actively interacts with the environment and receives an immediate reward for its actions. The learning objective is to maximize the overall reward over a number of

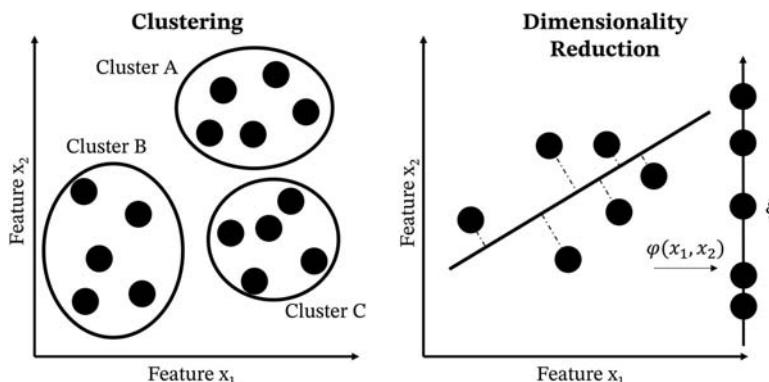


Figure 15.3 Unsupervised learning tasks: clustering and dimensionality reduction.

actions. Reinforcement learning has, for example, been used to teach computers how to play the game of Go, defeating the world champion [4]. A detailed introduction to reinforcement learning is given in Chapter 15 – Modern Spacecraft GNC—Reinforcement Learning (Fig. 15.4).

The following sections of this chapter will give an overview of the basic concepts and methods in ML which are necessary to understand modern DL-based AI applications. The first section describes the *k-means* algorithm for *clustering* as an example for unsupervised learning. Then, three ML models of stepwise increasing complexity are used to introduce techniques, concepts, and typical tasks of supervised learning: model representation, parameter learning, and generalization properties are described for *linear regression*. *Logistic regression* is then presented as a nonlinear model for *binary classification*.

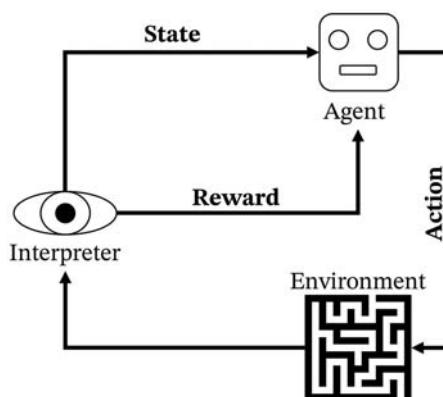


Figure 15.4 Reinforcement learning.

classification tasks and prepares the ground for the description of ANN. The *multilayer perceptron* (MLP), a basic DL model, is finally used to describe *multi-class classification* and the backpropagation learning algorithm for ANNs. Finally, CNNs and recurrent neural networks (RNNs) are introduced as modern DL architectures for computer vision (CV) and sequence tasks, respectively.

Unsupervised learning: k-means clustering

A common unsupervised learning problem is clustering, where the aim is to partition a given unlabeled dataset $D = (x^{(1)}, x^{(2)}, \dots, x^{(m)})$ of m samples $\mathbf{x} \in \mathbb{R}^d$ into a number of k coherent subsets or clusters $S = (S_1, S_2, \dots, S_k)$.

One of the most popular methods for clustering is the k-means algorithm [5], which aims at minimizing the within-class variance,

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^k \sum_{x \in S_i} \|x - \mu_i\|^2$$

where μ_i is the mean of all points in S_i . Beside the input dataset D , the number k of desired clusters has to be provided to the algorithm.

k-mean consists of two iterative steps, the cluster assignment step and the update step, preceded by an initialization step which is only executed once:

1. *Initialization*: k preliminary clusters are defined, for example, by randomly choosing k samples x as preliminary cluster means. All samples are then assigned to the cluster represented by the closest preliminary cluster mean, defined by the Euclidean distance.
2. In the *cluster assignment step*, the distance of each sample x to the number of k means μ of all clusters is calculated, and x is assigned to the cluster corresponding to the closest cluster mean.
3. In the *update step*, the cluster mean is updated taking into account the assignments from step 1.

Steps 1 and 2 are repeated until the cluster means and, therefore, the cluster assignments do not change any further. k-means is not guaranteed to find the optimal partition since it can get stuck in local optima, and the final result may depend on the particular initialization. For this reason, it is a common practice to execute k-means multiple times with varying starting conditions (Fig. 15.5).

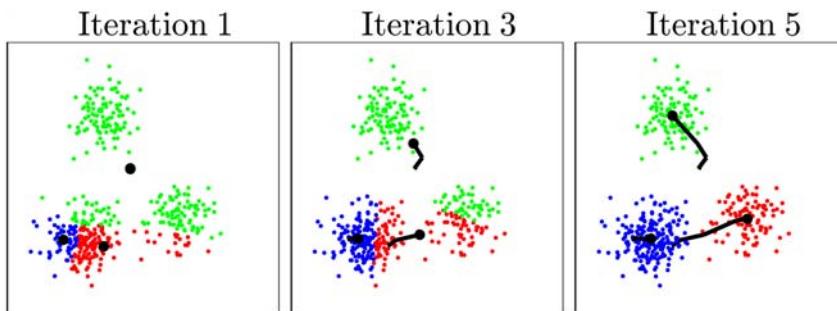


Figure 15.5 Three processing iterations of K-means clustering. Black stars mark the cluster means, and their position update is illustrated by the black tracks.

Supervised learning: regression and classification

Linear regression

Model representation

Linear regression is the simplest regression algorithm and a classic example for the concepts and notation involved in supervised learning. Given a training data set of m samples:

$$D = ((\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)}))$$

where for each input vector $\mathbf{x} \in \mathbb{R}^N$, the corresponding output $y \in \mathbb{R}$ is included and a family of linear hypotheses $h_w(\mathbf{x}) : X \rightarrow Y$,

$$h_w(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \dots + w_Nx_N \quad (15.1)$$

where each particular hypothesis is defined by a parameter vector $\mathbf{w} \in \mathbb{R}^{N+1}$, the task of linear regression is it to find the hypothesis which best describes the dataset D .

Cost function for regression

This corresponds to finding the parameter values w for which the error between the given ground truth (GT) values y and the model predictions $\hat{y} = h(\mathbf{x})$ is minimized. The error for a given training example $(\mathbf{x}^{(i)}, y^{(i)})$ is given by the *loss function*. For linear regression, it is defined by the squared error:

$$L^{(i)}(w_0, w_1) = (h_w(x^{(i)}) - y^{(i)})^2$$

The performance on the whole training set is calculated by the *cost function*, also called the *error function* or *objective function*, which for linear

regression is the average error over the whole data set, defined by the mean squared error (MSE) $J(w) = \frac{1}{2m} \sum_{i=1}^m (h_w(x^{(i)}) - y^{(i)})^2$.

By using the squared error, it is guaranteed that the result is always positive regardless of the sign of the predicted and target values. While the choice of the cost function depends on the problem at hand, MSE loss works well for most (not only linear) regression problems.

The objective is to find the hypothesis, i.e., the model parameters w , which minimize $J(w)$:

$$\underset{\mathbf{w}}{\operatorname{argmin}} J(\mathbf{w})$$

Parameter learning: gradient descent

Although the optimization problem for linear regression with MSE can be solved analytically using the normal equation, here we will introduce and focus on *gradient descent* (GD), a more general numerical approach which forms the basis for parameter learning in many other ML models, including ANNs and DL problems. Nevertheless, there exist several alternatives for numerical optimization methods, such as:

- Newton and Gauss–Newton methods.
- Levenberg–Marquardt algorithm.
- Conjugate gradient.

The detailed description of each method goes beyond the scope of this book, but it is important to remark that the methods have common basis and, above all, similar implementations. Indeed, the Levenberg–Marquardt curve-fitting algorithm can be regarded as a combination of GD method and Gauss–Newton. The behavior of Levenberg–Marquardt algorithm resembles the GD method when the weights are far from their optimal values and acts more like the Gauss–Newton method when the weights are close to their optimal value.

Going back to GD, to illustrate the idea behind it, we will use the simplified case of $N = 1$ and $w_0 = 0$, for which the hypothesis reduces to $h_w(x) = w_1 x_1$ and can be represented by a straight line through the origin, with w_1 controlling the slope of the line. In this case, the cost function is a quadratic polynomial in w_1 , shown in Fig. 15.6.

GD uses the gradient $\nabla J(w) = \left(\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \dots, \frac{\partial J}{\partial w_N} \right)^T$ of the cost function with respect to the model parameters in order to systematically update the

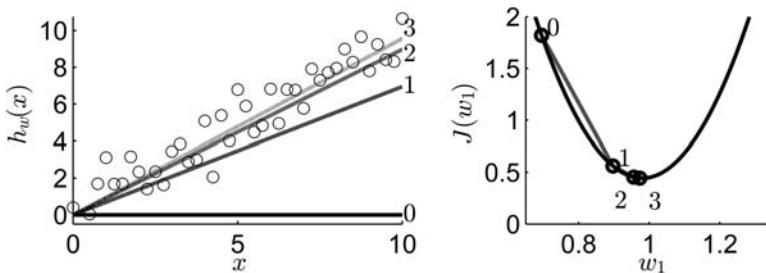


Figure 15.6 Linear regression and gradient descent.

parameter values and to gradually approach a minimum of the cost function. This is illustrated in Fig. 15.6 for the simplified hypothesis: starting from an initial value of parameter w_1 , GD determines an increment Δw_1 for which the error decreases, i.e., $J(w_1 + \Delta w_1) < J(w_1)$. This corresponds to moving down the slope of $J(w_1)$. Since the direction of the steepest positive slope is given by the partial derivative $\frac{\partial J}{\partial w_1}$, the update is done in the opposite direction, $w'_1 = w_1 - \eta \frac{\partial}{\partial w_1} J(w)$, where η is called the *learning rate* which determines the step width and is a hyperparameter which has to be chosen manually. This update is repeated until reaching a minimum of the cost function. For convex functions like the MSE cost function used in linear regression, GD reaches the global minimum. For different, nonconvex cost functions with local minima, this is, however, not guaranteed (Fig. 15.7).

In the general N -dimensional case, the update for w'_0 and w'_k ($k \geq 1$) is given by:

$$\begin{aligned} w'_0 &= w_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_w(x^{(i)}) - y^{(i)}) \\ w'_k &= w_k - \alpha \frac{1}{m} \sum_{i=1}^m ((h_w(x^{(i)}) - y^{(i)}) x^{(i)}) \end{aligned} \quad (15.2)$$

which, using the gradient, can be written as $w' = w - \alpha \nabla J(w)$.

Feature engineering and polynomial regression

The linear hypothesis in Eq. (15.1) can easily be extended to capture more complex, nonlinear problems through the addition of nonlinear features or feature combinations. For example, for a two-dimensional (2D) input $x = (x_1, x_2)$, we can define new features $x_3 = \sqrt{x_1}$ or $x_4 = x_1 x_2$ and add these

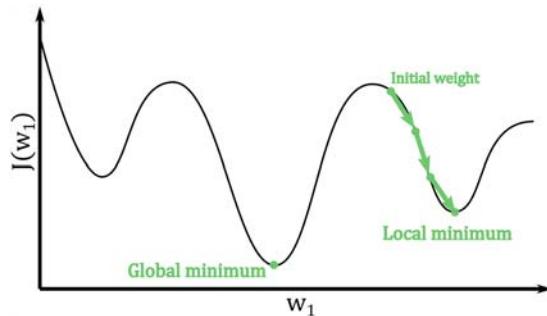


Figure 15.7 Gradient descent, local and global minima.

to our hypothesis. The manual process of creating new features is called *feature engineering* and ideally involves previous domain knowledge about the problem at hand. A common, systematic way of increasing model complexity are *polynomial features*, which use polynomial combinations of the features with degree less than or equal to the specified degree. For example, for a 2D input $x = (x_1, x_2)$, the two-degree polynomial features are $1, x_1, x_2, x_1x_2, x_1^2, x_2^2$, resulting in the hypothesis:

$$h_w(x) = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4x_1^2 + w_5x_2^2 \quad (15.3)$$

Generalization: under- and overfitting, training, test, and validation set

During training, optimization is performed with the objective to reduce the error on the *training set*. It is, however, not guaranteed that a model which works well on the training data will perform equally well on new data. Instead, a *test set* of previously unseen data is used to assess the final performance and generalization of the trained model. In order to make valid statements on the generalization capabilities, train and test sets have to meet certain properties: they have to be independent from each other, meaning that they cannot share data points; and it should be safe to assume that both data sets are drawn from the same probability distribution.

By comparing the error on the training set with the test (also called generalization) error, it can be determined if the model generalizes well, or if it is under- or overfitting the data (Fig. 15.8):

- *Underfitting*. Both the training and test errors are high, as the model does not account for relevant information present in the training set. The model is said to have *high bias/low variance* because the assumptions the model is based on are too rigid and prevent it from capturing the variance in the data.

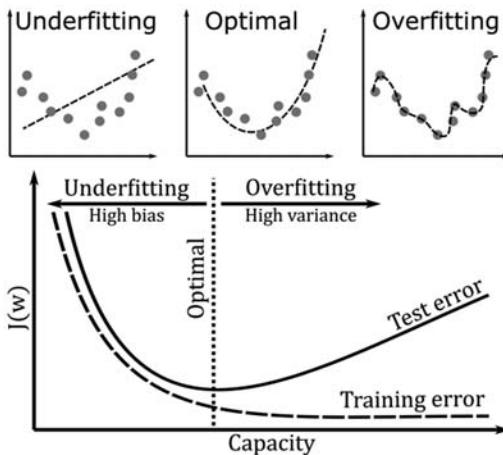


Figure 15.8 Relationship between model capacity and error.

- *Overfitting*. The gap between test and training error is large. The model learns properties and patterns which are very specific to the training data, leading to a small training error. It, however, fails to generalize to the unseen test data, yielding a large test error. This corresponds to *low bias/high variance*.

The propensity to over- or underfit is captured by the *model capacity*, which can be loosely defined as a model's ability to approximate complex problems and fit a variety of functions. In the polynomial regression example above, the model capacity increases with the degree of the added polynomial features: the higher the degree, the more variation can be captured by the model. In general, in order for a model to perform well, its capacity has to match the complexity of the specific task: a low-capacity model is unable to solve a complex problem (Fig. 15.8, top left), while on the other hand, a high-capacity model may fit the specific data too closely (Fig. 15.8, top right). This property of the model is called the *bias-variance trade-off* (Fig. 15.8, bottom) [6].

In addition to the trainable parameters or weights, a model is further defined by its *hyperparameters*: parameters which are not inferred during model training but are concerned with the learning algorithm's behavior (e.g., the learning rate α for GD) or the model selection, like the type or structure of the model (e.g., the degree in polynomial regression). Just as the training set has to be separate from the test set, a distinct *validation set* has to be used to assess the performance of the model during the tuning of the hyperparameters in order to guarantee that the model generalizes well to unseen data.

Logistic regression for binary classification

Model representation

In classification problems, the target variable or class label y which is to be predicted has a discrete value used to distinguish between two (binary classification) or more (multiclass classification) classes. Classical examples for classification problems include sentiment analysis, anomaly detection (e.g., fraud detection), or image classification (e.g., distinguish between cats and dogs, classification of handwritten digits). Binary classification can also be used for fault detection, where given some sensor output x , the objective is to determine if a spacecraft is working correctly ($y = 0$) or deviates from its nominal performance ($y = 1$).

Despite the slightly misleading name, *logistic regression* is not used for regression problems, but is one of the most popular and basic ML algorithms for classification. The name merely originates from its similarities to linear regression and from the use of the logistic function. Instead of directly calculating the binary class (0,1) for each input, the idea behind logistic regression is to (1) predict the conditional probability $P(y|x, w)$ of label y being 1 given the input features x and the model parameters w and (2) applying a threshold to the probability value in order to predict the discrete class label, i.e.,

$$\hat{y} = \begin{cases} 1 & \text{if } P(y=1|x, w) \geq 0.5 \\ 0 & \text{if } P(y=1|x, w) < 0.5 \end{cases}$$

In order to obtain probability values between 0 and 1, logistic regression uses the linear hypothesis seen for linear regression,

$$z = w_0 + w_1x_1 + w_2x_2 + \dots + w_Nx_N, \quad (15.4)$$

and applies the sigmoid function (also called logistic function, hence the name logistic regression). The sigmoid function (Fig. 15.12) is defined as $\sigma(z) = \frac{1}{1+e^{-z}}$ and maps any real value into the range 0–1. The model hypothesis for logistic regression is thus given by:

$$h_w(x) = \sigma(w_0 + w_1x_1 + w_2x_2 + \dots + w_Nx_N) = \sigma(z) \quad (15.5)$$

By defining an additional constant feature $x_0 = 1$, logistic regression can be presented in a vectorized formulation. Given the n-dimensional feature vector $\mathbf{x} = [x_0 \ x_1 \ \dots \ x_n]^T$ and the parameter or weight vector $\mathbf{w} = [w_0 \ w_1 \ \dots \ w_n]^T$, the input to the logistic function, $z = w_0 + w_1x_1 + \dots + w_nx_n$, can be vectorized to $z = \mathbf{w}^T\mathbf{x}$, and the model equation for logistic regression can be written as:

$$h_w(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}). \quad (15.6)$$

In order to illustrate the idea behind logistic regression, note that $\sigma(z) \geq 0.5$ for $z \geq 0$ and $\sigma(z) < 0.5$ for $z < 0$, and thus:

$$\hat{y} = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

In a 2D ($N = 2$) binary classification example, this means that any example with features x_1 and x_2 which satisfy the equation $w_0 + w_1x_1 + w_2x_2 \geq 0$ will result in a hypothesis prediction $\hat{y} = 1$. The straight line defined by $w_0 + w_1x_1 + w_2x_2 = 0$ is called the *decision boundary* as it separates the two regions for which $h_w(\mathbf{x})$ predicts either class 0 or 1. In higher dimensional problems, the decision boundary is formed by a hyperplane.

More complex models with nonlinear decision boundaries can be generated by adding polynomial features analogous to the approach seen earlier for polynomial regression. For example, by adding quadratic terms to Eq. (15.5),

$$h_w(\mathbf{x}) = \sigma(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2)$$

the decision boundary can now take on more complex shapes. For illustration, parameter values $w_0 = -1$, $w_1 = w_2 = 0$, $w_3 = w_4 = 1$ result in a decision boundary defined by the circle equation $x_1^2 + x_2^2 = 1$ shown in Fig. 15.9. More complex decision boundaries are possible adding higher degree polynomial features.

The decision boundary defined above is no property of the data set but of the model hypothesis, and hence of parameters which have to be learned from the training data. As seen for linear regression, this is done by solving an optimization problem involving an adequate loss function.

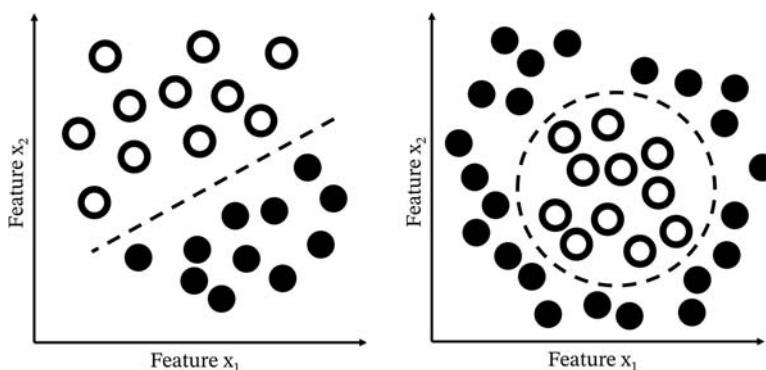


Figure 15.9 Classification with linear and nonlinear decision boundary.

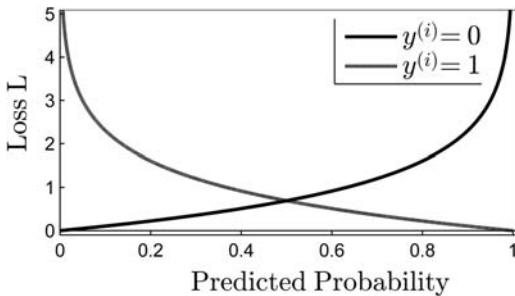


Figure 15.10 Binary cross-entropy loss.

Cost function for binary classification

Because the nonlinearity of the logistic regression hypothesis $h_w(\mathbf{x})$ renders the MSE cost function used for linear regression to be nonlinear and non-convex, the *binary cross-entropy* loss function is used instead, given by:

$$L(h_w(\mathbf{x}^{(i)}), y^{(i)}) = \begin{cases} -\log(h_w(\mathbf{x}^{(i)})), & y^{(i)} = 1 \\ -\log(1 - h_w(\mathbf{x}^{(i)})), & y^{(i)} = 0 \end{cases} \quad (15.7)$$

In the case of true label $y^{(i)} = 1$, L goes to infinity for $h_w \rightarrow 0$, strongly penalizing an incorrect prediction, whereas the loss disappears ($L = 0$) for $h_w = 1$. Analogously, penalization is inverted for $y^{(i)} = 0$, as shown in Fig. 15.10.

As for linear regression, the complete cost function is defined as the sum over all training samples. For that purpose, function Eq. (15.7) can be simplified into a single equation:

$$J(\mathbf{w}) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_w(\mathbf{x}^{(i)})) + (1 - y^{(i)}) \log(1 - h_w(\mathbf{x}^{(i)}))] \quad (15.8)$$

By multiplying the two logarithmic terms by $y^{(i)}$ and $(1 - y^{(i)})$, respectively, only the one corresponding to the specific true class label $y^{(i)}$ will add to the total cost. The binary cross-entropy loss is one of the standard loss functions used for classification problems.

In contrast to the MSE loss function for linear regression, there is no closed form to determine the optimal parameters using Eq. (15.8), and numerical methods like GD have to be used. The GD update equations for logistic regression are the same as Eq. (15.2) for linear regression, where

h_w now represents the hypothesis for logistic regression. Logistic regression models tend to overfit the data, particularly in high-dimensional settings. For this reason, regularization methods are often used to prevent the model from fitting too closely to the training data.

Artificial neural networks for multiclass classification

ANNs are the state-of-the-art for many ML problems. They are able to learn complex nonlinear hypotheses, for example, in nonlinear classification problems. One of the strengths of ANNs lies in their capability to automatically learn abstract feature representations of the input data, removing the need for manual feature engineering.

As seen in the previous sections, manual feature generation like the addition of polynomial features is a common and effective way to increase model complexity. However, in addition to increasing the tendency for overfitting, in practice, this method might not always be feasible, especially for problems which already come with a big number of input features, since depending on the order of polynomials to add the total number of features may increase drastically. Just by considering the addition of second-degree polynomial terms, the number of features increases roughly quadratically. Depending on the number of initial features and the data to process, this can become computationally infeasible.

As we will see in more detail discussing networks topologies, this is especially limiting for CV problems like image classification, where the input consists of whole images, or more precisely, the input features are given by the intensity values of each image pixel. For an image of size 100×100 , including only second-order polynomial features will result in millions of features, and it is not guaranteed that the increase in complexity will be sufficient. As we will see, ANNs can deal with a large input feature space by successively learning more and more abstract features automatically.

Universal approximation theorem

The founding theorem that proves the capability of ANNs to automatically learn abstract feature representations of the input data is the so-called universal approximation theorem. The classical form reads:

Let $\phi : \mathcal{R} \rightarrow \mathcal{R}$ be a nonconstant, bounded, and continuous function (called the activation function). Let I_m denote the m -dimensional unit hypercube $[0, 1]^m$. The space of real-valued continuous functions on I_m is denoted by $C(I_m)$. Then, given any $\epsilon > 0$ and any function $f \in C(I_m)$, there exists an integer N , real constants $v_i, b_i \in \mathcal{R}$, and real vectors $w_i \in R^m$, $\forall i = 1, \dots, N$, such that we may define:

$$F(\mathbf{x}) = \sum_{i=1}^N v_i \phi(\mathbf{w}_i^T \mathbf{x} + b_i)$$

as an approximate realization of the function f , that is:

$$|F(\mathbf{x}) - f(\mathbf{x})| < \varepsilon, \forall \mathbf{x} \in I_m$$

Roughly speaking, the theorem states that there is always a neural network architecture (number of layers, weights, and biases) to approximate a given function to a desired accuracy.

Model representation

The name *artificial neuronal network* derives from the fact that it consists of networks of interconnected nodes, analogous to biological neurons in the brain. The neurons in an ANN are organized in layers. In the classic ANN model called MLP, neurons belonging to the same layer receive inputs from all the neurons of the previous layer and send their output to each neuron of the following layer (Fig. 15.11). The MLP is a *feedforward* ANN—the flow from layer to layer always goes in the same direction—and it is *fully connected*, since every neuron is connected to every neuron in the next layer.

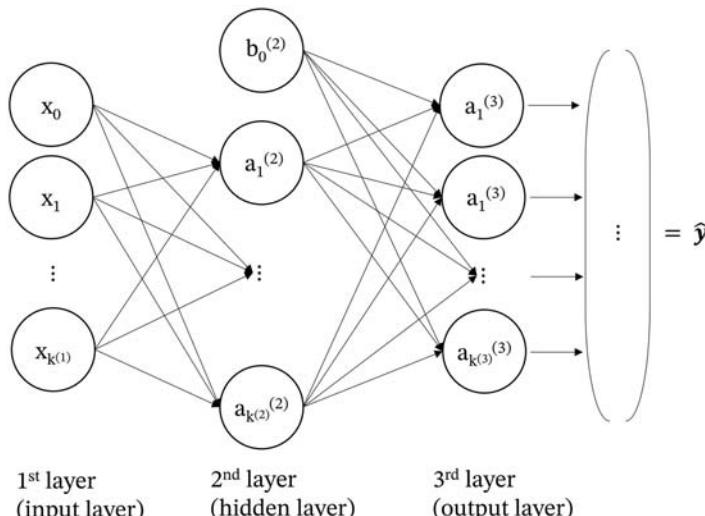


Figure 15.11 Multilayer perceptron/Fully Connected Neural Network with three layers. The $b_0^{(2)}$ represents the bias of the second layer (often referred as first activation neuron).

The first layer of an ANN, which consists of the input features, is called the *input layer*, and accordingly, the last layer, which returns the final results, is called the *output layer*. All the intermediate layers are *hidden layers*, since in contrast to the input and output layer, where (in the case of supervised learning) we can compare the value of each node with its true value given in the training set, the true values for hidden layers are unknown. When counting the number of layers, the input layer is commonly not taken into account, and therefore an ANN with one input, one hidden, and one output layer is referred to as two-layer network. An ANN with one or more hidden layers is called a DNN. Analogously, a network without hidden layer is called a *shallow neural network*.

Fig. 15.11 shows a two-layer MLP for binary classification as an example, with the input layer representing $k^{(1)}$ input features, one hidden layer with $k^{(2)}$ units, and $k^{(3)}$ units in the output layer for the classification output. The parameters characterizing a MLP are the following:

- N - The number of layers or *depth* of the network.
- $k^{(l)}$ - The number of units in layer l .
- $g(z)$ - The nonlinear activation function applied to each unit (e.g., logistic function $\sigma(z)$).
- $w_i^{(l)}$ - The $k^{(l-1)}$ -dimensional parameter or weight vector of unit i in layer l with elements $w_{ij}^{(l)}$.
- $b_i^{(l)}$ - The bias parameter of unit i in layer l .
- $z_i^{(l)}$ - The $k^{(l)}$ -dimensional input vector to unit i in layer l .
- $a_i^{(l)}$ - Activation, i.e., scalar output of unit i in layer l . All outputs of layer l are summarized in the $k^{(l)}$ -dimensional activation vector $a^{(l)}$. To facilitate a generalized notation, we define $a^{(0)} = x$ and $a^{(L)} = y$.

The input vector z to each unit is calculated from the vector of activations of all units of the previous layer,

$$z_i^{(l)} = w_i^{(l)} a^{(l-1)} + b_i^{(l)}$$

The activations of unit i in layer j are then calculated applying the activation function:

$$a_i^{(l)} = g(z_i^{(l)})$$

With respect to Eq. (15.5), here the logistic function σ has been substituted by the more general nonlinear activation function $g(z)$. If the logistic

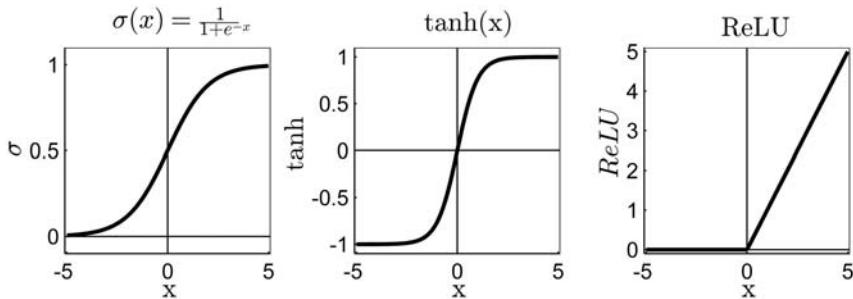


Figure 15.12 Common activation function.

function is used as an activation function, each unit acts like a logistic regressor with inputs being the activations of all units of the previous layer. Hence, logistic regression can be viewed as a shallow, one-layer neural network. In practice, for ANNs, the logistic function σ is often replaced by different nonlinear *activation functions* (Fig. 15.12). Some of the most common functions are:

- The *sigmoid* or *logistic function* (Fig. 15.12, left) maps values to the range $(0, 1)$ and is the default activation function when the output is interpreted as a probability, e.g., the probability of corresponding to a given class in a classification task.
- The *hyperbolic tangent* (Fig. 15.12, center) has a shape similar to the sigmoid function, but is zero-centered, returning values in the range between -1 and 1 . It is also used in binary classification tasks.
- The *rectified linear unit* (*ReLU*) activation function (Fig. 15.12, right) has become an extremely popular activation function for hidden units in DNNs and especially for CNNs, due to a series of advantages: compared to the sigmoid function, computation is more efficient, it provides better gradient propagation, and mitigates the problem of vanishing gradients which will be discussed later.
- The *softmax* activation function can be seen as a generalization of the logistic function to multiple dimensions and is defined as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

It is commonly used in the output layer of a multiclass classification neural network in order to normalize the output to a probability distribution over the predicted output classes since it guarantees that the sum over all class probabilities adds up to 1.

The notation of the neural network equations can be further simplified by defining the weight matrix $W^{(l)}$ containing all the weights which define the connections between layer l and $l - 1$. Thus, column i of matrix $W^{(l)}$ corresponds to the parameter vector $w_i^{(l)}$ which contains the weights connecting all units in layer $l - 1$ with unit i in layer l . The equations summarizing all operations in layer l can then be written as:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)T} \mathbf{a}^{(l-1)} + b^{(l)}$$

$$\mathbf{a}^{(l)} = g(\mathbf{z}^{(l)})$$

$\mathbf{W}^{(l)}$ has dimensions $(k^{(l-1)} + 1) \times k^{(l)}$, where “+1” corresponds to the bias unit which is added to the dimension of layer $l - 1$.

Cost function for multiclass classification

As mentioned before, the choice of the loss function depends on the specific ML problem. In the example of linear regression, the mean square error was used; for binary classification using logistic regression, binary cross entropy was introduced. For the ANN, we will use multiclass classification with the categorical cross entropy as an example.

Where in binary classification the model prediction was restricted to two classes, labeled 0 or 1, in multiclass classification, the output labels correspond to K classes, where $K \geq 3$. The neural network represents these classes by K output units, resulting in a K -dimensional output vector $\hat{\mathbf{y}} = h_w(\mathbf{x}) \in \mathbb{R}^K$. Using the softmax activation function in the output layer, the sum over all vector elements is 1, and thus each element \mathbf{y}_k can be interpreted as the probability of belonging to class k .

The *categorical cross-entropy* cost function for this case is calculated from the GT labels \mathbf{y} and the predicted values $\hat{\mathbf{y}}$ and is defined as:

$$J(\mathbf{y}, \hat{\mathbf{y}}, \mathbf{W}) = -\frac{1}{m} \sum_{t=1}^m \sum_{k=1}^K [y_k^{(t)} \log(\hat{y}_k^{(t)}) + (1 - y_k^{(t)}) \log(1 - \hat{y}_k^{(t)})]$$

where the subscript k indicates the k -th vector component and t indicates the training sample. In other words, the total categorical cross-entropy cost is obtained by summing over the binary cross-entropy cost of all K output units.

ANN parameter learning: backpropagation and gradient descent

Training of an ANN follows the same steps as seen earlier for linear and logistic regression: firstly, it requires the definition of a cost function which matches the problem in question and defines the optimization objective, followed by the optimization procedure itself, e.g., using GD. The main difference regarding ANNs is the increased complexity of the computation steps involved. The objective is to find the weights \mathbf{W} which minimize the cost function, $\underset{\mathbf{W}}{\operatorname{argmin}} J(\mathbf{y}, \hat{\mathbf{y}}, \mathbf{W})$, which can be achieved through iterating the following steps:

1. For a given training sample $(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})$, perform the model computations to obtain output $\hat{\mathbf{y}}$. For a feedforward ANN, this step is called *forward propagation* since calculations are performed layer by layer and information is propagated forward through the network.
2. Calculate the scalar loss or error $J(\mathbf{y}, \hat{\mathbf{y}}, \mathbf{W})$.
3. Use GD to update the model weights. To this end, we need to calculate the changes in the total error connected to changes in each single weight of the network, given by the partial derivatives $\frac{\partial}{\partial w_{ij}^{(l)}} J(\mathbf{y}, \hat{\mathbf{y}}, \mathbf{W})$, where

$w_{ij}^{(l)}$ denotes the weight between neuron j of the previous layer $l-1$ and neuron i of the current layer l .

The last step poses the biggest challenge when it comes neuronal networks. The deep structure of an ANN means that the output or activation of each hidden unit does affect many units in the downstream layers, and therefore contributes to the total cost through many separate paths. In order to combine all these effects, an efficient and systematic method for the computation of the partial derivatives is required. The development of the *backward propagation* (or short, *backprop*) algorithm as an efficient method to calculate these loss derivatives was one of the major reasons for the increasing interest in ANNs in the 1980s [7]. Backward propagation calculates the loss derivatives for all hidden units with respect to their activation a , from where it is easy to get the loss derivatives of the weights going into the hidden unit.

In the forward pass, calculations follow the following schematic sequence:

$$x \rightarrow z^{(1)} \rightarrow a^{(1)} = \sigma(z^{(1)}) \rightarrow z^{(2)} \rightarrow a^{(2)} \rightarrow \dots \rightarrow y = a^{(L)} \rightarrow J(y, w)$$

In backpropagation, this order is reversed, and the changes of the error function with respect to the unit output y of the last layer is calculated first, then with respect to the unit input of the last layer, from where the changes

of the error function with respect to the weights are obtained. From there, the process is repeated sequentially for the previous layers (where the unit outputs are the activations), until reaching the first hidden layer:

$$\begin{array}{ccccccc} & \frac{\partial J}{\partial z^{(L)}} & & \frac{\partial J}{\partial z^{(L-1)}} & & \frac{\partial J}{\partial z^{(1)}} & \\ \frac{\partial J}{\partial y} = \frac{\partial J}{\partial a^{(L)}} \rightarrow & \downarrow & \rightarrow \frac{\partial J}{\partial a^{(L-1)}} \rightarrow & \downarrow & \rightarrow \dots \rightarrow \frac{\partial J}{\partial a^{(1)}} \rightarrow & \downarrow & \\ & \frac{\partial J}{\partial w_{ij}^{(L)}} & & \frac{\partial J}{\partial w_{ij}^{(L-1)}} & & \frac{\partial J}{\partial w_{ij}^{(1)}} & \end{array}$$

In this way, backpropagation traces the separate contributions to the error starting at the back, i.e., at the output layer, passing through the network layer by layer.

Backpropagation is based on the repeated application of the chain rule of calculus, which states that for a real number x and two functions g and f which map from a real number to a real number and are linked by $y = g(x)$ and $z = f(g(x)) = f(y)$, the derivative of z with respect to x can be obtained by $\frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx}$. As a consequence, backpropagation requires both the loss function and the activation function to be differentiable.

The following example illustrates the calculations for any two units chosen from subsequent layers: the output layer l and a hidden layer $l-1$, indicated by the superscripts. The subscripts j and i run over all units of layer l and $l-1$, respectively.

The calculations involved are:

1. Apply the chain rule to calculate the derivative of the loss with respect to the total input received by unit j ,

$$\begin{aligned} \frac{\partial J}{\partial z_j^{(l)}} &= \frac{da_j^{(l)}}{dz_j^{(l)}} \frac{\partial J}{\partial a_j^{(l)}} \\ &= a_j^{(l)} \left(1 - a_j^{(l)}\right) \frac{\partial J}{\partial a_j^{(l)}} \end{aligned}$$

where we assumed that the derivative of the logistic unit $a = \sigma(z)$ is $\frac{da}{dz} = a(1-a)$. $\frac{\partial J}{\partial a_j^{(l)}}$ is the derivative of the loss with reference to the activation of unit j in layer l .

2. To get $\frac{\partial J}{\partial w_{ij}^{(l)}}$, we observe that $z_j^{(l)}$ is a linear function of the weights $w_{ij}^{(l)}$ and the outputs of the previous layer, $a_i^{(l-1)}$. Again, we apply the chain rule:

$$\frac{\partial J}{\partial w_{ij}^{(l)}} = \frac{\partial z_j^{(l)}}{\partial w_{ij}^{(l)}} \frac{\partial J}{\partial z_j^{(l)}} = a_i^{(l-1)} \frac{\partial J}{\partial z_j^{(l)}}$$

where $\frac{\partial J}{\partial z_j^{(l)}}$ has already been calculated in step 1.

This can be repeated for all hidden layers. In order to backpropagate from layer l to $l-1$, we substitute l by $l-1$ in step 1 and observe that we can determine the change of the error depending on the changes in the output of unit i in layer $l-1$ by summing over all outgoing connections of unit i :

$$\frac{\partial J}{\partial a_i^{(l-1)}} = \sum_j \frac{dz_j^{(l)}}{da_i^{(l-1)}} \frac{\partial J}{\partial z_j^{(l)}} = \sum_i w_{ij}^{(l)} \frac{\partial J}{\partial z_j^{(l)}}$$

Here, $\frac{dz_j^{(l)}}{da_i^{(l-1)}}$ characterizes the change of the total input to unit j with the changes in the output of unit i , which is simply the weight connecting unit i and j . The second term, $\frac{\partial J}{\partial z_j^{(l)}}$, is known from step 1. The weight update for GD is then given by:

$$\Delta w_{ij}^{(l)} = -\eta \cdot \frac{\partial J}{\partial w_{ij}^{(l)}} = -\eta \cdot a_i^{(l-1)} \frac{\partial J}{\partial z_j^{(l)}}$$

where η is the learning rate, a user-defined coefficient, as already noted.

Vanishing or *exploding gradients* pose a problem when training a DNN with gradient-based learning methods and backpropagation [8,9]. Since weights are updated proportionally to the partial derivative of the error function with respect to the current weight, the learning process can come to a rest if the gradient becomes very small (vanishing gradient) or become unstable for very large gradients (exploding gradient). In each layer, the weight updates are obtained from the gradients calculated from all later layers using the chain rule, effectively involving the multiplication of the gradients of those later layers. Thus, if gradients are smaller than 1, the resulting weight update gets smaller with each layer, leading to an exponential

decay of the weight updates and preventing any learning progress in the earlier layers. The opposite effect of an exponential increase of the weight update can occur for gradients greater than 1.

Various solutions such as gradient clipping (i.e., restricting maximum gradient values to a fixed threshold), weight regularization, and alternative or modified activation functions have been proposed and are used to prevent vanishing and exploding gradients [9]. In residual networks, skip connections pass gradient information from a deeper layer directly to a nonadjacent previous layer, thus helping maintain signal propagation even in deeper networks [10]. For RNNs, which are especially affected by the problem as we will see later in the chapter, special architectures like the long short-term memory (LSTM) have been developed to prevent vanishing gradients [11].

GD is used to train neural networks in an iterative manner, with weight updates performed through repeated forward-backpropagation passes of the available training samples. Depending on the specific application, computational capacity, and data availability, different training strategies are typically applied which differ in the number of samples in each update step.

In *batch learning* or *batch GD*, the weight update is only executed after all the input-target data have been presented to the network. One complete presentation of the training dataset is typically called an epoch. The forward and backward pass is performed one after each epoch. In batch learning, the system is not capable of continuous learning while doing. The training dataset consists of all the available data. This generally takes a lot of time and computational effort, given the typical dataset sizes. For this reason, batch learning is generally performed on-ground. The system that is trained with batch learning first learns offline and then is deployed and runs without updating itself: it just applies what it has learnt.

Mini-batch learning is a variation of full batch learning, where the update is performed on subsets of the complete dataset, and thus each epoch consists of a number of forward-backpropagation passes. Mini-batch GD is widely used when the number of samples in the training dataset is very big and it becomes infeasible to perform the computations needed for the parameter update, which involve the calculation and storage of the cost function and its gradients, on the whole set at once. *Stochastic gradient descent* (SGD) is a special case of mini-batch learning with a batch size of one sample.

In *incremental learning*, often referred to as *online learning*, the system is trained continuously as new data instances become available. They could be clustered in mini-batches or come as standalone datum using SGD. Online learning systems are tuned to set how fast they should adapt to incoming

data: typically, such parameter is again referred to as learning rate. A high learning rate means that the system reacts immediately to new data, by adapting itself quickly. However, a high learning rate means that the system will also tend to forget and replace the old data. On the other hand, a low learning rate makes the system stiffer, meaning that it will learn more slowly. Also, the system will be less sensitive to noise present in the new data or to mini-batches containing nonrepresentative data points, such as outliers. In addition, Lyapunov-based methods are very suitable for incremental learning due to their inherent stepwise trajectory evaluation of the stability of the learning rule.

Types of artificial neural networks

While the MLP represents the most basic type of neural network, based on one-dimensional (1D) input vectors and feedforward operations, other network architectures exist for specific applications. For instance, another popular feedforward network is the radial basis function neural networks (RBFNNs).

Two of the most important and widely used type of ANNs are CNNs for 2D input data, i.e., images, and RNNs, which are used for data representing sequences like time series or text sentences. CNN and RNN will be introduced in the following.

Radial basis function neural network

An RBFNN is a single-layer shallow network, whose neurons are Gaussian functions. Such network architecture possesses a quick learning process, which makes it suitable for online dynamics identification and reconstruction. The highlights of the mathematical expression of the RBFNN are reported here for clarity.

For a generic state input $\mathbf{x} \in R^n$, the components of the output vector $\gamma \in R^j$ of the network is:

$$\gamma_l(\mathbf{x}) = \sum_{i=1}^m w_{ij} \Phi_i(\mathbf{x})$$

In a compact form, the output of the network can be expressed as:

$$\gamma(\mathbf{x}) = \mathbf{W}^T \Phi(\mathbf{x})$$

where $\mathbf{W} = [w_{il}]$ for $i = 1, \dots, m$ and $l = 1, \dots, j$ is the trained weight matrix and $\Phi(\mathbf{x}) = [\Phi_1(\mathbf{x}) \Phi_2(\mathbf{x}) \dots \Phi_m(\mathbf{x})]^T$ is the vector containing the output of

the radial basis functions (RBFs), evaluated at the current system state. RBFNN is used in classification, function approximation, time series prediction problems, etc. The RBF network learns to designate the input to a center, and the output layer combines the outputs of the RBF and weight parameters to perform classification or inference. RBFs are suitable for classification, function approximation, and time series prediction problems. Typically, the RBF network has a simpler structure and a much faster training process with respect to MLP, due to the inherent capability of approximating nonlinear functions using shallow architecture. As one could note, the main difference of the RBFNN with respect to the MLP is that the kernel neuron is a nonlinear function of the information flow, instead of linear: in other words, the actual input to the layer is the nonlinear radial function $\Phi(\mathbf{x})$ evaluated at the input data \mathbf{x} , most commonly Gaussian ones. The most used radial-basis function that can be used and that are found in space applications are [12–14]:

- Gaussian $\Phi(r) = e^{-\frac{(r-c)^2}{2\sigma^2}}$.
- $\Phi(r) = \frac{1}{(\sigma^2+r^2)^\alpha}$.
- Linear $\Phi(r) = r$.
- Thin-plate spline $\Phi(r) = r^2 \ln(r)$.
- Logistic function $\Phi(r) = \frac{1}{1+e^{(r/\sigma^2)-\theta}}$.

where r is the distance from the origin, c is the center of the RBF, σ is a control parameter to tune the smoothness of the basis function, and θ is a generic bias. The number of neurons is application-dependent, and it shall be selected by trading-off the training time and approximation, especially for incremental learning applications. The same consideration holds for the parameters $\eta = \frac{1}{\sigma}$, which impacts the shape of the Gaussian functions. A high value for η sharpens the Gaussian bell shape, whereas a low value spreads it on the real space. On one hand, a narrow Gaussian function increases the responsiveness of the RBF network, on the other hand, in case of limited overlapping of the neuronal functions due to too narrow Gaussian bells, the output of the network vanishes. Hence, ideally, the parameter η is selected based on the order of magnitude of the exponential argument in the Gaussian function. The output of the neural network hidden layer, namely the radial functions evaluation, is normalized:

$$\Phi_{norm}(\mathbf{x}) = \frac{\Phi(\mathbf{x})}{\sum_{i=1}^m \Phi_i(\mathbf{x})}$$

The classic RBF network presents an inherent localized characteristic; whereas, the normalized RBF network exhibits good generalization properties, which decreases the curse of dimensionality that occurs with classic RBFNN.

Convolutional neural networks

CV is one of the areas which has benefited most from the recent advances in DL. Most of state-of-the-art techniques for CV tasks like image recognition or classification and object detection are based on DL approaches and form the basis for applications for autonomous navigation and self-driving cars or internet image searches. Typical CV problems are:

- *Image classification.* It determines if the input image contains a certain object.
- *Object detection.* It determines the position of one or several objects in the input image, see [Fig. 15.13](#).
- *Image segmentation.* It partitions the input image into different regions, e.g., foreground and background regions, see [Fig. 15.14A and B](#) (cfr. [Chapter 9 – Navigation](#)).

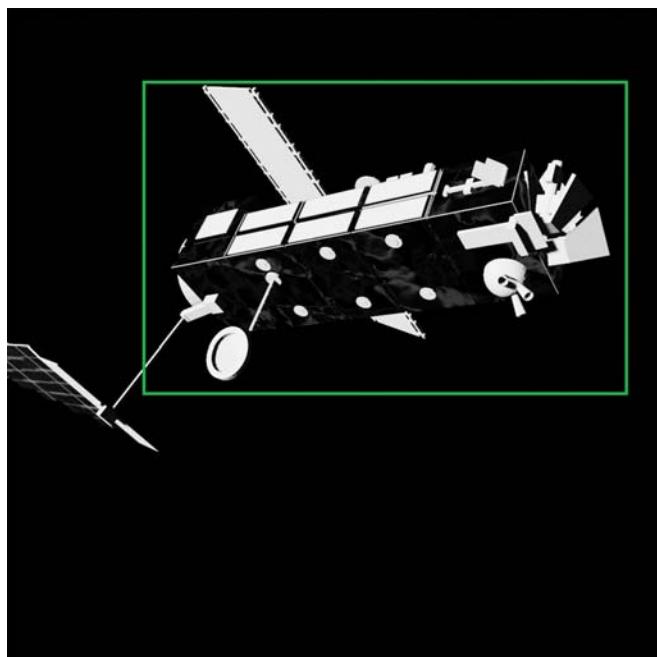


Figure 15.13 Object detection example. A bounding box indicates the position of the detected body satellite.

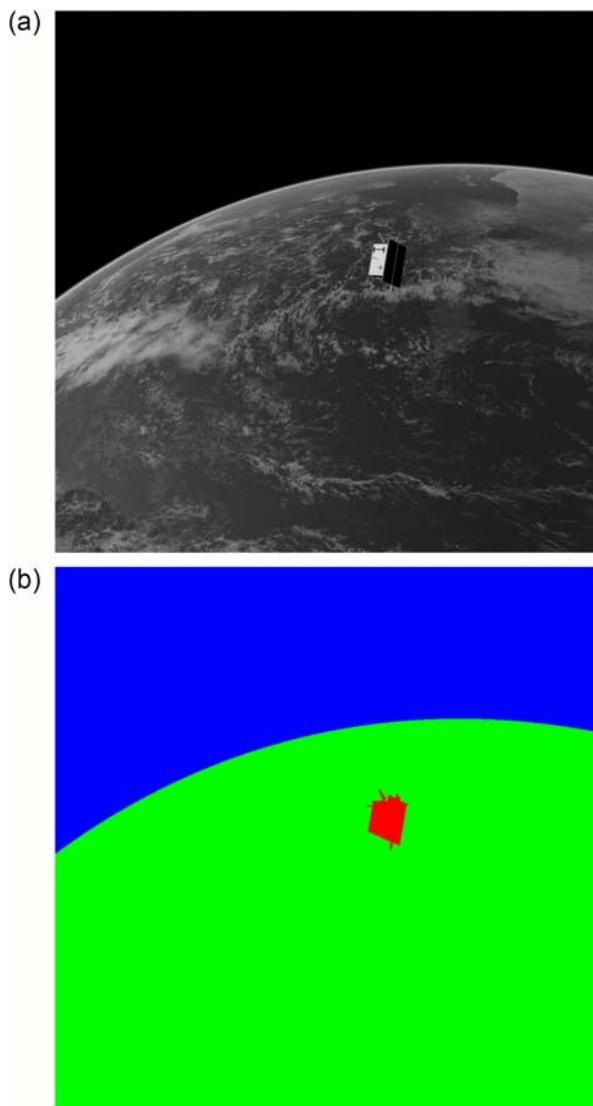


Figure 15.14 Image segmentation example. Colors indicate different entities of the image. Red: spacecraft; blue: deep space; green: Earth. Refer to Ref. [15] for a comprehensive dataset for training deep learning models.

One of the challenges of CV problems is the size of the input features and the number of parameters involved, which can make the application of a fully connected neural network, like the MLP to an image classification task, infeasible. For example, a three channel RGB image of size

1000×1000 can be flattened into a 1D input vector with three million ($1000 \times 1000 \times 3$) elements. Feeding this input feature into a neural network with 1000 hidden units in the first layer will result in a weight matrix with three billion parameters. A neural network of this size will not only be prone to overfitting, but it will also impose unrealistic computational and memory requirements.

In CNNs, this problem is solved using convolution operations, which are the fundamental building blocks of CNNs. In the following, we will first define the convolution operation, give an example of how convolution filters can extract image features, and finally show how they are used in a CNN.

In the context of CNNs, the convolution for a given 2D input image I and a 2D filter (also called *kernel*) K is given by²:

$$C(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

where $*$ is the convolution operator, and m and n run over all values that lead to legal subscripts of $I(i-m, j-n)$ and $K(m, n)$. Indices i and j refer to the pixel position in the input image, while indices m and n refer to the filter. The result is again a 2D output image, in the context of CNNs commonly called *feature map*, whose height and width depend on the size of the input image and the filter, as well as additional hyperparameters to be discussed below. The operation is illustrated in Fig. 15.15 for a 6×6 input image and a 3×3 filter; the filter is applied to the input image as a sliding window, starting at the top left corner of the input image. Overlapping filter and input

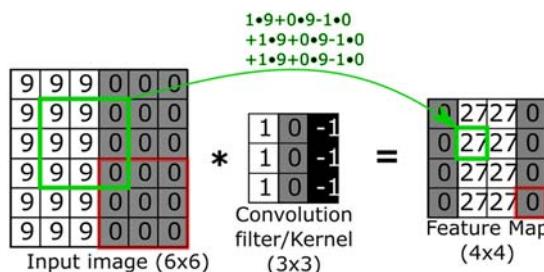


Figure 15.15 Convolution for vertical edge detection.

² Note that in the field of Machine Learning, often the related cross-correlation function $C(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$ is implemented and called *convolution*. For a detailed explanation see Ref. [17].

indices are multiplied element-wise, and the sum of the resulting nine values is stored at indices (1,1) of the output image. The next element, (2,1), is obtained by repeating the calculation after sliding the filter one pixel downwards, and analogously for all elements of the output image.

[Fig. 15.15](#) illustrates how convolution filters are able to extract visual features like vertical edges. The vertical edge in the input image, defined by the simultaneous drop from pixel intensity 9 to 0, results in high values/high intensities in the corresponding columns of the resulting *feature map*.

While the filter in our example was designed manually, one of the strengths of DNNs is the ability to learn the filter parameters automatically, and to extract more and more abstract representations of the input data with every hidden layer. In the context of CV tasks, these levels of abstraction correspond to the detection of simple visual features like edges/corners or blobs in the first layer, to parts of objects in the next, to complete objects the deeper we get into the neural network [16], as illustrated in [Fig. 15.16](#).

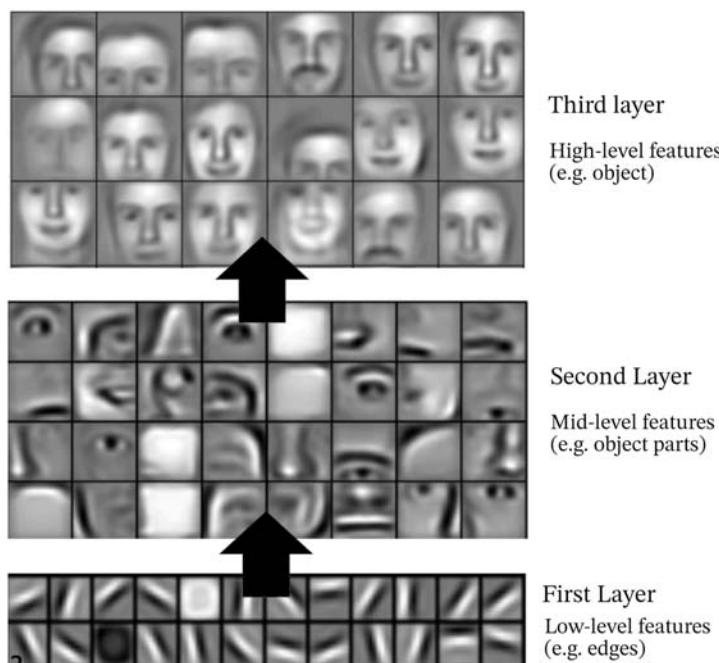


Figure 15.16 High- and low-level features. Adapted from H. Lee, R. Grosse, R. Ranganath, A.Y. Ng, *Unsupervised learning of hierarchical representations with convolutional deep belief networks*, Communications of the ACM, 54 (10) (2011), 95–103. <https://doi.org/10.1145/2001269.2001295>; M.Z. Asghar, M. Abbas, K. Zeeshan, P. Kotilainen, T. Hämäläinen, *Assessment of deep learning methodology for self-organizing 5G networks*, Applied Sciences 9 (15) (2019) 2975. <https://doi.org/10.3390/app9152975>.

The output dimension of a $f \times f$ convolution filter applied to a $n \times n$ input image is given by $k = (n-f+1) \times (n-f+1)$ (starting at the upper right corner of the image, the filter can be shifted $(n-f+1)$ -times in each dimension before reaching the right/bottom edge of the image). This means that the output of the operation is smaller than the original input, which can become problematic when convolutions are applied sequentially like in a multilayered neural network. In addition, pixels close to the corner and edges of the input image enter less calculations than those at the center. These issues can be avoided by *padding* the input image, i.e., by adding a fixed number p of rows/columns to each side of the input image before performing the convolution. These added rows and columns can contain a fixed constant value (*constant padding*, with the special case of *zero padding*), but they can also replicate or mirror the image values at the edges (*replication* and *mirror padding*). In order to determine the number of rows and columns to be added, a common padding scheme, called *same convolution*, uses $p = \frac{f-1}{2}$ to produce an output image of the same size as the input. Commonly, convolution without padding ($p = 0$) is called *valid convolution*. Since the filter size f is generally chosen to be an odd number, p takes on integer values.

A second parameter which controls image convolution is the *stride* s , which defines the number of pixels by which the filter is shifted after each convolution operation. For example, with $s = 2$, the filter will move from column (or row) 1 to 3, omitting column (row) 2. Thus, the stride roughly decreases the output size by a factor $\frac{1}{s}$ in each dimension. The exact formula for the output dimension including padding p and stride s is given by:

$$k = \left\lceil \frac{n + 2p - f}{s} + 1 \right\rceil \times \left\lceil \frac{n + 2p - f}{s} + 1 \right\rceil$$

Here, the floor operator \lceil is used to account for cases in which the fraction is not an integer value. Both the stride and the padding parameters are hyperparameters of a CNN and thus are chosen when deciding the specific neural network architecture.

The convolution operation is not limited to gray-scale images but can easily be generalized to images with multiple channels like RGB images. An input image of height h and width w is then defined by a three-dimensional matrix of size $h \times w \times c$, where c defines the number of channels, e.g., $c = 3$ for a three-channel RGB image. The convolution filter also becomes three-dimensional with size $f \times f \times c$, where the number of

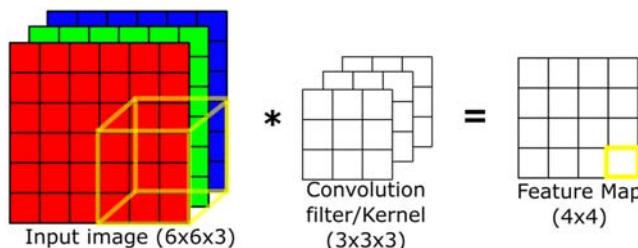


Figure 15.17 Convolution over volumes.

channels c is the same for the input image and the filter. As illustrated in Fig. 15.17, convolution is then performed by moving the three-dimensional filter over the width and height of the input image, just as for the 2D case, multiplying the overlapping image and filter indices, which now include indices corresponding to all channels, and finally summing up all the $f \cdot f \cdot c$ obtained values. Like for the case of 2D input image and filter, the result is a 2D feature map.

In a CNN, convolution filters represent the equivalent of the weight matrices of a fully connected neural network, containing the parameters learned during training. In general, each layer of a CNN is made up of multiple such filters each of which can extract different image features and generate a corresponding feature map, also called activation map. The final output of each filter is obtained by the addition of a bias parameter and element-wise application of the nonlinear activation function, with the ReLU activation being a common choice in CNNs. If the layer contains F filters, the output of the layer is the F generated feature maps, stacked together in a $f \times f \times F$ array (Fig. 15.18). Each layer can thus be characterized by:

- The width and height of the convolution filters.

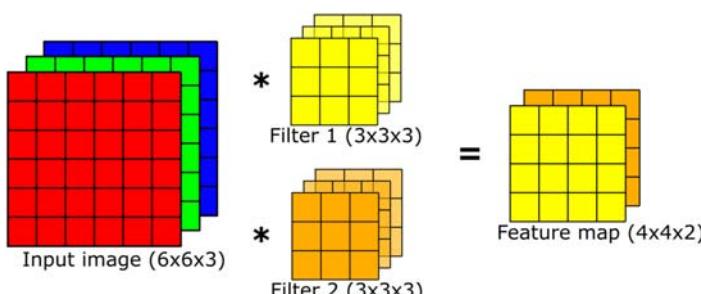


Figure 15.18 The results from multiple filters are stacked.

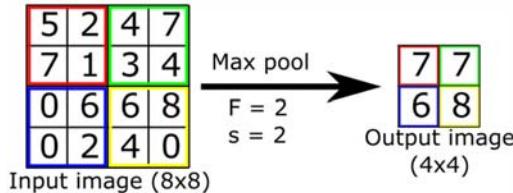


Figure 15.19 Max pooling.

- The number of input channels and output channels. The number of channels of a layer must match the number of channels of its input.
- The hyperparameters of the convolution operation: padding and stride.
In addition to convolution layers, a CNN may also include other types of neural network layers, the most common being *pooling layers* and *fully connected layers*.

Pooling layers are often introduced after a convolutional layer in order to reduce the number of parameters by downsampling the output of the previous convolution layer before passing it to the next. It can also have the effect of making the detected features more robust. The most common form of pooling in CNNs is *max pooling*: the max pooling filter of size F is moved over the input image or feature map with a step width defined by the stride s , and for each $F \times F$ filter location, only the maximum value of the input is retained in the output map (Fig. 15.19). The stride s is commonly set equal to the filter size, $s = F$. However, overlapping pooling where $s < F$ is also possible. *Average pooling* instead calculates the average value for the respective patches of the feature map.

In contrast to the convolution filters, padding is usually not applied for the pooling operation. Pooling is done independently for each input channel, meaning that the number of channels of the pooling output is the same as for the input. Since pooling is done using fixed operations (*max* or *mean*), pooling layers do not add additional trainable parameters to the network.

Fully connected layers can be included in a CNN if the task objective requires it, for example, when the final output is a single numeric value or a 1D vector, as is the case in image classification. If present, they are therefore usually added as the last layers of the network. The 1D input vector for a fully connected layer which immediately follows a CNN or pooling layer is typically created by flattening the 2D feature map.

Parameter sharing and sparsity of connections. As mentioned at the beginning of the section, one of the advantages of CNNs when compared to fully

connected networks is the reduction in the total number of parameters involved. The output of a convolutional network layer with three 5×5 -dimensional filters which takes a $64 \times 64 \times 3$ -dimensional input image will have dimensions $(64 - 5 + 1) \times (64 - 5 + 1) \times 3 = 60 \times 60 \times 3$. While in this situation, a fully connected layer taking in the $64 \times 64 \times 3 = 12288$ input features and having an equivalent number of $60 \times 60 \times 3 = 3600$ units will result in a weight matrix containing $3600 \times 12288 = 44236800$ parameters, the convolutional layer will contain only $(5 \times 5 + 1) \times 3 = 78$ parameters, independent of the size of the input image. This reduction in the number of parameters can be attributed to two characteristics of CNNs: *parameter sharing* and *sparsity of connections*. Parameter sharing greatly reduces the number of parameters by using the same filter in all the regions of the image (using the same filter while sliding over the image), instead of learning a different set of parameters for every location. Sparsity of connections means that in a convolutional layer, each input feature is generally only used in a number of $f \times f$ calculations. In contrast, in a dense, fully connected layer, the input to each hidden unit is calculated from all the input features. Differently to fully connected neural networks, where the size of the weight matrices depends on both the number of input features and the number of units in the layer, in a CNN, the number of parameters in the filters does not depend on the input dimension.

In the following, we will give a short overview of how the presented operations are used to construct neural networks for the CV tasks of image classification, image segmentation, and object detection.

Image classification networks

One of the classic and most influential DNN architectures to combine the basic elements of CNNs such as convolutions, max pooling, and fully connected layer is AlexNet [19]. In 2012, AlexNet achieved ground-breaking performances for image classification in the ImageNet LSVRC-2012 challenge, boosting the use of DL techniques for CV and serving as a starting point for many modern developments.

The architecture of this classification network is shown in Fig. 15.20, detailing the filter parameters and resulting sizes of the feature maps. AlexNet consists of five convolutional layers for feature extraction and three fully connected layers responsible for the classification. The first two convolutional layers are followed by overlapping max pooling, reducing the height and width of the resulting feature maps. While no pooling is performed

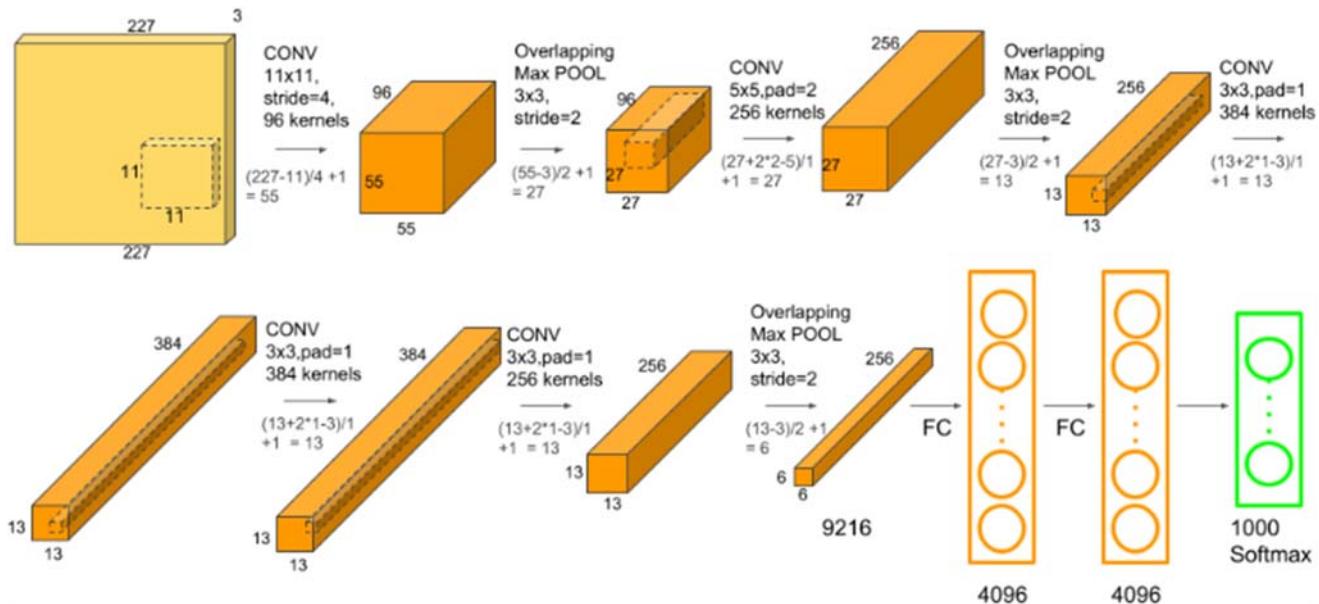


Figure 15.20 AlexNet architecture.

between the third, fourth, and fifth convolutional layers, the fifth convolutional layer is followed by a max pooling layer, the output of which goes into a series of two fully connected layers. The output layer, containing 1000 units corresponding to the 1000 class labels in the ILSVRC-2012 dataset [20], uses the softmax function to calculate the class probabilities. ReLU nonlinearity is applied after all the convolution and fully connected layers.

Image segmentation networks

Image segmentation can be thought of as a pixel-wise classification task. In *semantic segmentation*, each image pixel is assigned a corresponding class, thus labeling, for example, if a pixel belongs to a certain object in the image, the background or the foreground. *Instance segmentation* goes one step further by determining not only the pixel class but also different instances or objects of the same class in the image. For example, where the task of semantic segmentation is to determine if a certain pixel belongs to any satellite, instance segmentation in addition distinguishes to which of multiple satellites present in the image the pixel belongs. Since image segmentation makes predictions for each pixel of the input image, the output is a *segmentation map* with a one-to-one correspondence between pixels in the input image and in the segmentation map.

A popular approach for semantic image segmentation are fully convolutional models, meaning that they only contain convolutional layers and additional up-/downsampling layers (e.g., pooling layers), and which are often based on an encoder/decoder structure [21]: the encoder network downsamples the spatial resolution of the input and creates lower-resolution feature mappings which facilitate the classification task; the decoder network then upsamples the feature representations into a full-resolution segmentation map. Additional shortcuts, called *skip connections*, transfer information from layers of the encoder network to later layers of the decoder network, skipping some of the downsampling and upsampling steps in order to recover location detail which is lost during the sampling steps.

Fig. 15.21 illustrates the approach for a very popular segmentation network architecture, the U-Net [22]. The name originates from the symmetrical u-shaped structure of the network, where the left, descending path represents the encoder subnetwork while the right ascending path corresponds to the decoder network. The encoder follows the typical architecture

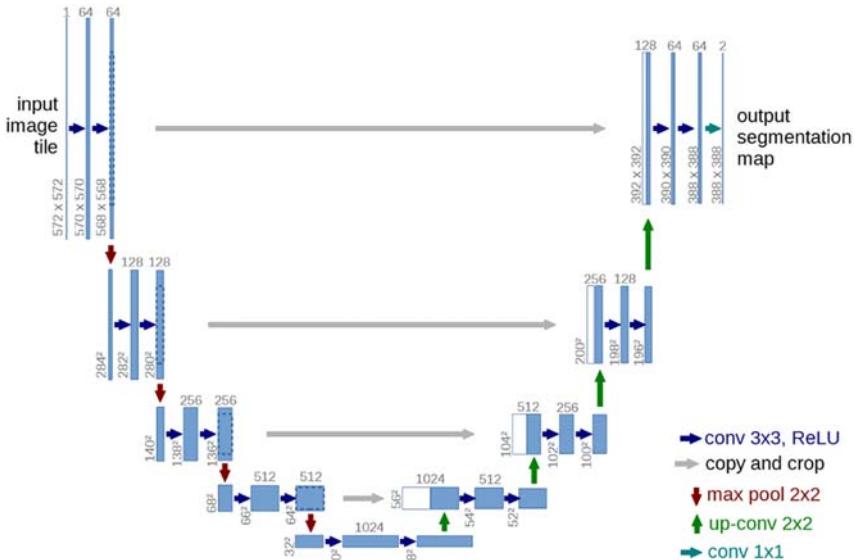


Figure 15.21 The original U-net architecture for image segmentation from Ref. [22] (example for 32×32 pixels in the lowest resolution). Each blue box corresponds to a multichannel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. *Adapted from Y. Cai, L. Qian, Y. Fan, L. Zhang, H. Huang, X. Ding, An automatic trough line identification method based on improved UNet, Atmospheric Research, 264 (2021), 105839, ISSN 0169-8095. <https://doi.org/10.1016/j.atmosres.2021.105839>.*

of a convolutional network, repeatedly applying combined convolution-pooling blocks consisting of two 3×3 convolutions, followed by a ReLU activation function and a 2×2 max pooling filter, yielding feature maps of less and less resolution but a higher number of channels. Conversely, every step in the decoder path consists of an upsampling convolution of the feature map, application of the skip connection by concatenation of the upsampled feature map with the corresponding feature map from the down-sampling path, a 2×2 convolution which halves the number of feature channels, and two 3×3 convolutions, each followed by a ReLU activation. The final layer uses a 1×1 convolution to reduce the 64-dimensional feature vector corresponding to each pixel to the desired number of classes. Note that due to the use of valid convolutions, the segmentation map is smaller than the input image and only contains the pixels for which the full context is available in the input image.

Object detection network

In object detection, the task is to determine the location and type of objects in a given image. For that purpose, object detectors return a bounding box for each detected object, alongside with the object category and the classification confidence. Common object detection models perform this task in three steps: region selection, feature extraction, and classification. While in principle this can be achieved by scanning the image using multiscale sliding windows and performing image classification on the corresponding image regions, this approach is computationally very expensive especially when based on DNN techniques, and more efficient methods are used which can be divided into two generic DL object detection frameworks [24]:

- The *region proposal-based framework* replicates the traditional pipeline by using two-stage detectors: in the first step, deep image features are extracted from the whole image and used to find approximate image regions. The second step then performs classification and bounding box regression on the proposed regions. Implementations of this approach include R-CNN [25], faster R-CNN [26], and mask R-CNN [27]. The R in the acronym stands for regional to highlight the segmentation approach this architecture uses to generate the output.
- The *regression/classification-based framework*, which includes one-stage detectors such as YOLO (“You Only Look Once”) [28], SSD (Single Shot MultiBox Detector) [29], and RetinaNet [30] that predict object bounding boxes without the region proposal step.

While region proposal-based techniques typically perform better than the regression/classification-based methods, the latter are computationally more efficient. In the following, the YOLO one-stage detector will be used as an example to illustrate the functioning of DL-based object detection (Fig. 15.22).

The idea behind YOLO is to simultaneously predict multiple bounding boxes and their respective class probabilities using a single CNN. First, YOLO divides the input image into an $S \times S$ grid. For each grid cell, it then predicts a number of B bounding boxes, each of which is defined by five values: the x and y coordinates which represent the center of the box relative to the grid cell, the width w and height h of the box, and the confidence score. In addition, C class probabilities are predicted for each grid cell. The output is encoded in an $S \times S \times (B * 5 + C)$ -dimensional tensor.

In the YOLO network, shown in Fig. 15.23 for parameters $S = 7$, $B = 2$, and $C = 20$, feature extraction is performed by 24 convolutional layers

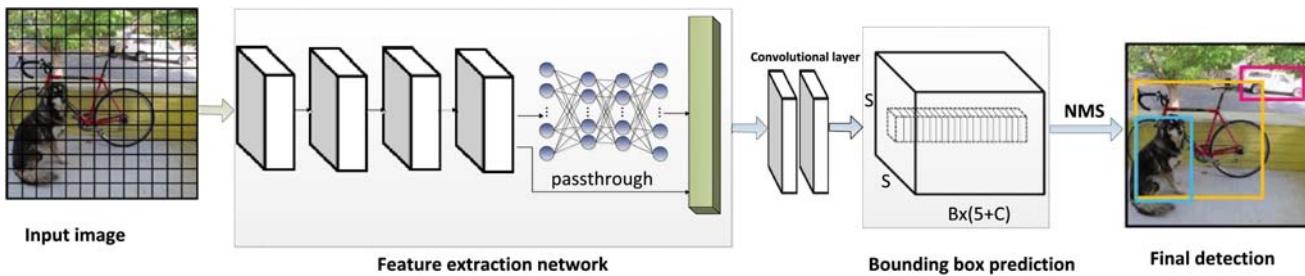


Figure 15.22 YOLO divides the image into a grid, and then for each grid cell predicts a predefined number of bounding boxes, as well as a confidence score for each of those boxes, and the class probabilities [28]. Adapted from Y. Yin, H. Li, W. Fu. Faster-YOLO: an accurate and faster object detection method, *Digital Signal Processing*, 102 (2020), 102756. <https://doi.org/10.1016/j.dsp.2020.102756>.

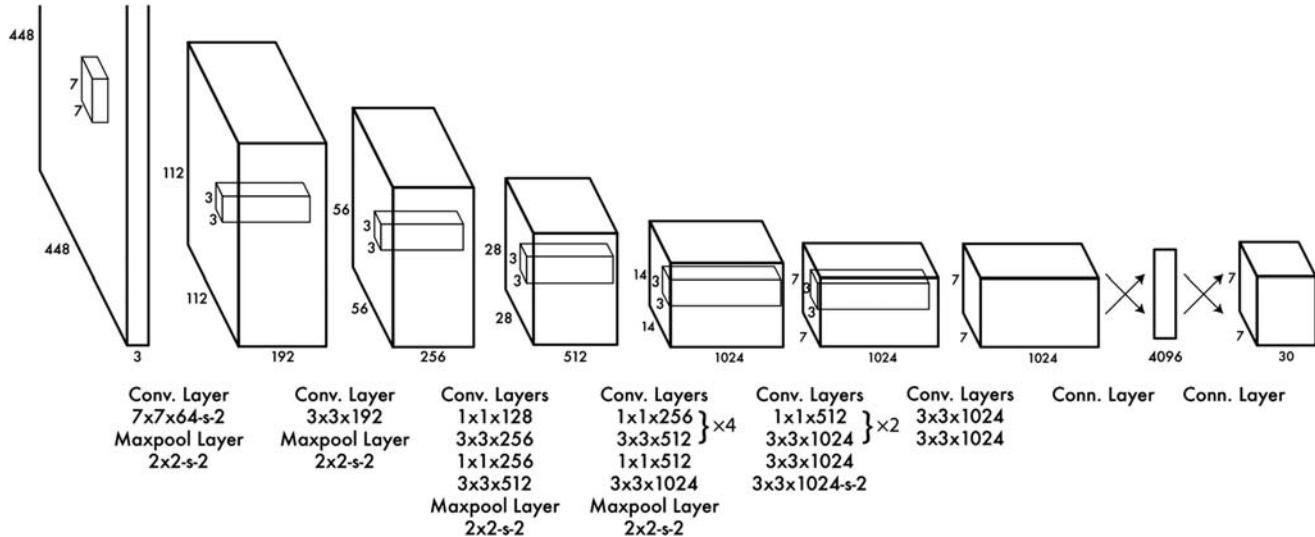


Figure 15.23 Architecture of the original YOLO Object detection network [28]. Adapted from H. Maudi Lathifah et al., *Fast and accurate fish classification from underwater video using you only look once*, IOP Conference Series: Materials Science and Engineering 982 (2020) 012003.

followed by two fully connected layers. 1×1 convolutional layers are used to reduce the feature space from preceding layers. Fully connected layers then use the extracted features to predict the output probabilities and bounding box coordinates. The final result of the network is the $7 \times 7 \times 30$ output tensor.

Recurrent neural networks

We have seen how feedforward, fully connected neural networks can be used to solve classification and regression tasks involving fixed-size input vectors, and how CNNs tackle complex image-related tasks. Both approaches, however, are insufficient to tackle problems involving timeseries and sequence data, which require the ability to model temporal dynamics and long-term interactions. Sequence data are ubiquitous and come in a great variety of forms, from audio and video data, to written sentences and DNA sequences.

In ML, tasks involving sequences are commonly categorized by the length of the input and output sequences (Fig. 15.24):

- *Many-to-one*. It maps a sequence of inputs to a single output. This scheme can be used, for example, for the classification of sequences such as sentiment classification, which determines if an input sentence is a positive or negative expression.
- *One-to-many*. It generates an output sequence given a single input. A possible application is the automatic generation of image captions, where a single image is mapped to an output sentence.
- *Many-to-many*. An input sequence is mapped to an output sequence of the same size. For example, in video classification or video activity recognition, each frame of the image sequence is mapped to a corresponding label.

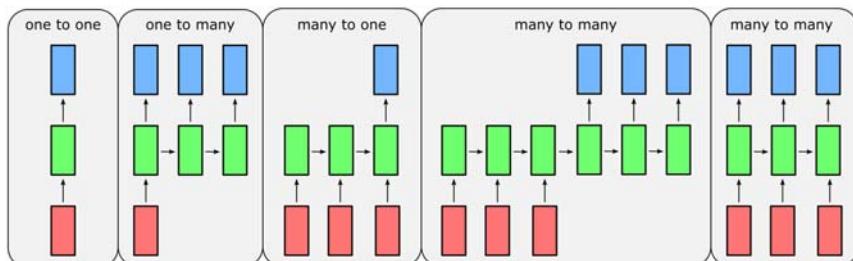


Figure 15.24 Sequence tasks.

- *Many-to-many*, using an encoder–decoder structure for sequence-to-sequence learning. This scheme is used for machine translation where an input sentence in the original language is first encoded and then decoded into the destination language. The original and translated sentence do not have to be of the same length.
- *One-to-one*. If both the input and output sequences have length one, inputs can be considered independently without any interactions. This scheme includes, for example, MLP-based classification.

In order to process sequence data efficiently, at each timestep, information from past (and possibly future) timesteps has to be available. RNNs tackle this problem by passing the activation from one timestep to the next. Let us consider a task where given an input sequence consisting of input vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t$, we want to predict the output sequence $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_t$. In an RNN, at time step t , the input vector \mathbf{x}_t is passed to a hidden network layer and the activations of the layer's units and the predicted output value \mathbf{y}_t are calculated. In contrast to a feedforward network, however, the network layer also receives the *hidden state* vector \mathbf{h}_{t-1} containing the activations of the previous timestep as an additional input. The updated hidden state \mathbf{h}_t and the output \mathbf{y}_t are obtained as:

$$\mathbf{h}_t = \sigma_h(\mathbf{W}_x \mathbf{x}_t + \mathbf{U}_h \mathbf{h}_{t-1} + \mathbf{b}_h)$$

$$\mathbf{y}_t = \sigma_y(\mathbf{W}_y \mathbf{h}_t + \mathbf{b}_y)$$

where $\mathbf{W}_y, \mathbf{W}_h, \mathbf{U}_h$ are parameter matrices, $\mathbf{b}_h, \mathbf{b}_y$ bias parameter vectors, and σ_h and σ_y activation functions. The hidden state is initialized as $h_0 = 0$.

RNNs are often shown in the form of a compressed diagram in Fig. 15.25 left, where the feedback of the hidden state vector is depicted as a recurrent connection. The involved forward computations can also be

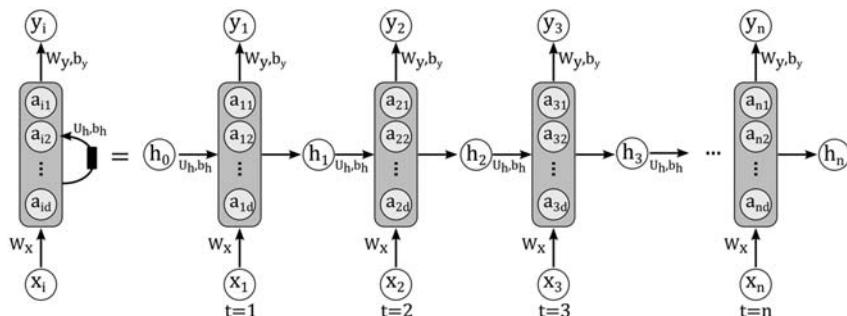


Figure 15.25 Diagram for a one-unit recurrent neural network. Compressed diagram on the left and the unfold version of it on the right.

represented for each timestep separately, or *unfolded* in time, making explicit the input, hidden state, and output at each step (Fig. 15.25, right). Through the recurrent link, the output at each timestep effectively depends on all the previous inputs $\mathbf{x}_{t'}$ (with $t' \leq t$). Since the same parameters (matrices $\mathbf{W}_y, \mathbf{W}_h, \mathbf{U}_h$ and biases $\mathbf{b}_h, \mathbf{b}_y$) are used at each time step, RNNs can be viewed as very deep feedforward networks where each timestep corresponds to one layer and all the layers share the same weights.

Backpropagation *through time* [33–35] is used to compute the derivatives of the total error with respect to all the states \mathbf{h} and all the parameters in the unfolded RNN analogously to a feedforward network, and GD is used for optimization.

Although they have been very successfully used to model complex dynamic systems, training RNNs has proved to be difficult due to the problem of *vanishing or exploding gradients* described earlier. Since the backpropagated gradients either grow or shrink at each time step, over many time steps, they can become very large (explode) or tend toward zero (vanish).

Nonlinear autoregressive exogenous model

The nonlinear autoregressive exogenous (NARX) model is a simple recurrent structure that uses the feedback coming from the output layer. The NARX model is based on the linear autoregressive exogenous (ARX) model, which is commonly used in time-series modeling. The defining equation for the NARX model is:

$$\mathbf{y}_k = \mathcal{N}(\mathbf{y}_{k-1}, \mathbf{y}_{k-2}, \dots, \mathbf{y}_{k-n}, \mathbf{u}_{k-1}, \mathbf{u}_{k-2}, \dots, \mathbf{u}_{k-n})$$

where \mathbf{y} is the network output and \mathbf{u} is the exogenous input, as shown in Fig. 15.26. Basically, it means that the next value of the dependent output signal \mathbf{y} is regressed on previous values of the output signal and previous values of an independent (exogenous) input signal. It is important to remark that, for a one step delay NARX, the defining equation takes the form of an autonomous dynamical system.

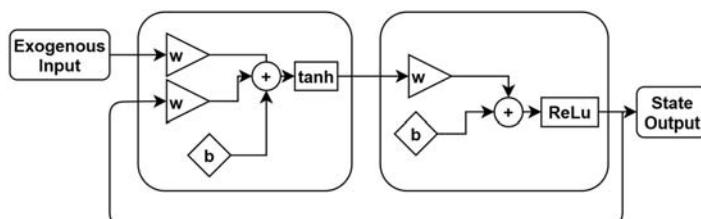


Figure 15.26 Schematic of the nonlinear autoregressive exogenous model.

Hopfield neural networks

The formulation of the Hopfield neural network (HNN) is due to Hopfield [36], but the formulation by Abe [37] is reportedly the most suited for combinatorial optimization problems, which is of great interest in the space domain. For this reason, here the most recent architecture is reported. A schematic of the network architecture is shown in Fig. 15.27.

In synthesis, the dynamics of the i -th out of N neurons is written as:

$$\frac{dp_i}{dt} = \sum_{j=1}^N w_{ij}s_j - b_i$$

where p_i is the total input of the i -th neuron, w_{ij} and b_i are parameters corresponding, respectively, to the synaptic efficiency associated with the connection from neuron j to neuron i , and the bias of the neuron i . The term s_i is basically the equivalent of the activation function:

$$s_i = c_i \tanh \frac{p_i}{\beta}$$

where $\beta > 0$ is a user-defined coefficient, while c_i is the user-defined amplitude of the activation function.

The recurrent structure of the network entails a dynamic of the neurons; hence, it would be more correct to refer to $p(t)$ and $s(t)$, function of time or any other independent variable.

An important property of the network, which will be further discussed in the application for parameter identification, is that Lyapunov stability

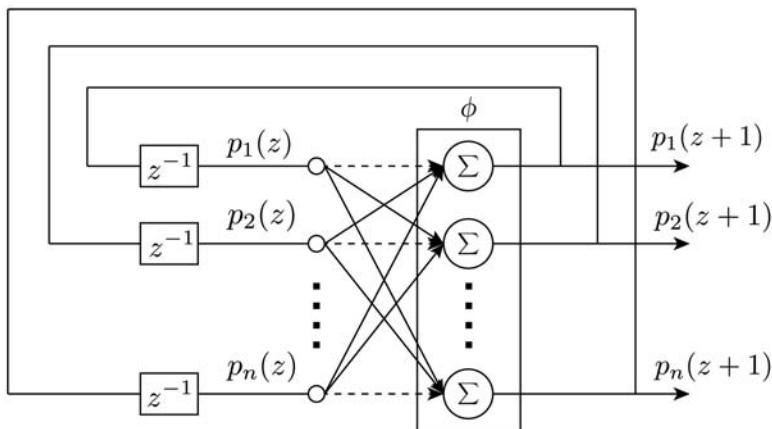


Figure 15.27 Schematic of the Hopfield neural network.

theorem can be used to guarantee its stability. Indeed, since a Lyapunov function exists, the only possible long-term behavior of the neurons is to asymptotically approach a point that belongs to the set of fixed points, meaning where $\frac{d\mathcal{V}}{dt} = 0$, being \mathcal{V} the Lyapunov function of the system, in the form:

$$\mathcal{V} = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} s_i s_j + \sum_{i=1}^N b_i s_i = -\frac{1}{2} \mathbf{s}^T \mathbf{W} \mathbf{s} + \mathbf{s}^T \mathbf{b}$$

where the right-hand term is expressed in compact form, with s vector of s neuron states, \mathbf{b} the bias vector. A remarkable property of the network, in fact, is that trajectories always remain within the hypercube $]-c_i, c_i[$ as long as the initial values belong to the hypercube too [38]. For implementation purposes, the discrete version of the HNN is typically employed [39,40].

Long short-term memory

In order to solve the issues of vanishing gradients, more complex RNN variants have been developed. One of the first and widely used architecture is the LSTM network [11]. An LSTM uses a special hidden unit able to store inputs over many time steps and acting as long-time memory. This memory cell is connected to itself at the next timestep, thus copying its own real-valued state and accumulating the external signal. Another hidden unit, the *forget gate*, which learns to decide when to clear the content of the memory, is used as a multiplicatively gate of this self-connection.

The equations describing the LSTM implement this idea using the activation vector of the *forget gate* \mathbf{f}_t ; the activation vector of the input/*update gate* \mathbf{i}_t ; the activation vector of *output gate* \mathbf{o}_t ; the cell *input activation vector* $\tilde{\mathbf{c}}_t$; and the *cell state vector* \mathbf{c}_t :

$$\mathbf{f}_t = \sigma_g(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f)$$

$$\mathbf{i}_t = \sigma_g(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i)$$

$$\mathbf{o}_t = \sigma_g(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o)$$

$$\tilde{\mathbf{c}}_t = \sigma_c(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \sigma_c(\mathbf{c}_t)$$

Here, σ_g and σ_c are the sigmoid and hyperbolic tangent activation function, and the operator \circ denotes the element-wise product. The initial values are $h_0 = 0$ and $c_0 = 0$. Each of the gates \mathbf{f}_t , \mathbf{i}_t and \mathbf{o}_t learns its own set of parameters calculated from the input \mathbf{x}_t and the previous hidden state \mathbf{h}_{t-1} . The use of the sigmoid activation function in the calculation of the gate vectors means that their element values will lie within the range $[0,1]$ and will often be approximately 0 or 1, thus either blocking the gate or letting the corresponding input elements pass. The forget and update gates decide how and which elements of the previous cell state vector \mathbf{c}_{t-1} and the input activation vector or “candidate cell state” $\tilde{\mathbf{c}}_t$ enter the updated cell state \mathbf{c}_t . Finally, the new activation or hidden state \mathbf{h}_t is calculated from the updated cell state, filtered by output gate \mathbf{o}_t . Thanks to the gating mechanism, which allows the LSTM to let the cell state vector \mathbf{c}_t pass unaffected over many timesteps, information can be stored over a long time. LSTM computations are illustrated in Fig. 15.28.

For learning very complex functions, various RNNs can be stacked together to form deep RNNs. Each layer in such a network has its own set of parameters, while all units of the same layer (in the unrolled view of the network) share the same parameters. The hidden state $\mathbf{h}_t^{(l)}$ in layer l at timestep t is then calculated not only from the hidden state at the previous timestep, $\mathbf{h}_{t-1}^{(l)}$, but also from the hidden state from the layer below, $\mathbf{h}_t^{(l-1)}$. Due to the temporal dimension, training only one RNN layer can already be computationally expensive. For this reason, deep RNNs remain rather shallow in practice when compared to feedforward neural networks, which very often have a great number of layers.

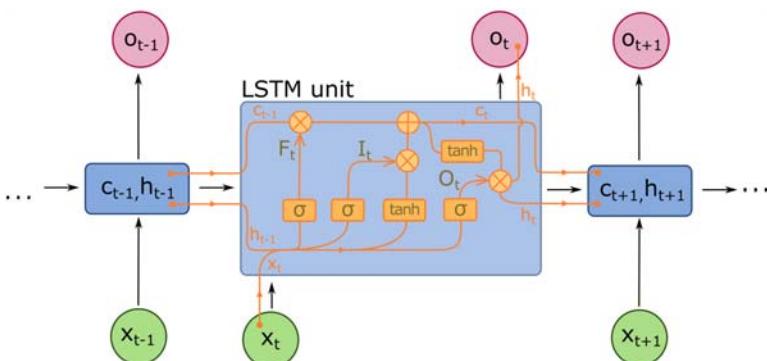


Figure 15.28 Schematic illustration of the LSTM unit.

Applications scenarios and AI challenges

The section presented a wide range of theoretical background that is essential for a space engineer who would like to dive into the AI domain. As remarked throughout the discussion, the described methods are valid across different domains; hence, it is quite hard to delineate a clear correspondence between methods and specific space-related applications.

Most of the embedded systems have been designed based on linear algebra and linearization. This is true also for the space technology. Although the linear design has served our purposes very well in the past, it imposes constraints and limitations on the potential of current technology for more demanding space missions.

Nature and generally universe has a nonlinear behavior. Putting linear systems into a nonlinear environment requires a lot of effort and resources from the engineers to tune it. As we have seen, a lot of concepts upon stability, observability, etc., requires the knowledge of linearized system and, above all, they guarantee the properties of the real system only within the validity of the linearization.

Nonlinear systems increase the capabilities of the technology, and new attributes emerge from its implementation. Of course, attention is required to enhance the capabilities that serve the objectives of the mission without inserting useless complications. Autonomous spacecraft GNC systems can truly benefit from this approach. Certainly, among those, navigation and control can be enhanced from online deployment of AI-based systems. Indeed, the demand of highly accurate navigation and control in a nonlinear and unpredictable harsh environment including time constraints compels the designers to consider alternative methodologies and concepts. One of these ideas is to use adaptive systems with nonlinear elements. AI can be thought as a particular kind of nonlinear adaptive system. The way it can be adopted is strongly dependent on the technology level and the current knowledge. For instance, AI techniques could allow the spacecraft to achieve highly accurate relative navigation when arriving to an unknown target. The most interesting application in that field are proximity operations around an asteroid or uncooperative target, relative pose estimation around an uncontrolled satellite, pinpoint landing on the asteroid or the Moon.

To perform online ML from a controls perspective toward a model-based ML setting, one needs to be able to construct models from data via system identification. The GNC system relies on mathematical model of the environment to calculate the estimated states and planned control.

The accuracy and adaptivity of the algorithms are pursued by implementing online learning based on AI techniques. This allows the agent to learn the surrounding environment as it flies, refining its mathematical representation on-board. The refinement increases the accuracy of the model, on which GNC is synthesized on-board, but also it captures variation of the environment, enhancing the adaptivity of the algorithm. The system identification modeling accuracy when used in a closed-loop setting needs the understanding of the trade-off between the number of samples needed to build reliable models and the performance degradation due to coarse modeling.

However, there are some unresolved challenges to apply AI algorithms on GNC systems, for instance, inadequate data sets for training, the theoretical understanding and modeling of the behavior of the AI system, the generalization of the learned features to different scenarios, and its validation. If the verification of such system is covered by existing European Cooperation for Space Standardization (ECSS) standards (as in Refs. [41–44]), their validation is much more complex. It is then important that the selection of an AI technique shall consider the objective of the qualification of the complete system, especially in the context of critical application.

Another challenge is related to the need of AI techniques in terms of processing and memory. If the processing and memory resources available on-board are constantly increasing, the AI techniques may still need some optimization to be executed on-board spacecraft.

Finally, we can frame two additional limitations of the current status of AI and ML for space applications:

- The trade-off between adaptivity and robustness in the design of the GNC system. On one hand, we are trying to design ML systems that evolve continuously by learning via the interaction with the dynamical and physical environment. On the other hand, they should pursue optimized solutions that are robust, explainable, and secure.
- The AI and ML algorithms borrowed from data science often lack in efficiency, robustness, and interpretation coming from a purely data-drive approach. The foundation of classical GNC theory instead lies in the mapping of physics into the model-based design concept.

In this section, we try to summarize guidelines and schematized typical approaches found in literature. In [Table 15.1](#), a summary of ANN types is presented, highlighting the most relevant field of application. Obviously, the table is not exhaustive and reports the common trend in the adoption of particular network types.

Table 15.1 Summary of most common ANN architecture used in space dynamics, guidance, navigation, and control domain. The training type is supervised (S), unsupervised (U), and reinforcement learning (R).

Network type	Architecture	Learning paradigm	Training algorithm	Space application
<i>Feedforward</i>	MLP	S/R	Backpropagation	Dynamics approximation, value-function approximation
	RBFNN	S/U/R	Backpropagation Lyapunov K-means clustering	Dynamics approximation, regression, time-series prediction
	AE	U	Backpropagation	Dimensionality reduction, state-space modeling, data encoding, anomaly detection
	CNN	S	Backpropagation	Feature detection, object detection, image classification, image-based regression, vision-based navigation
<i>Recurrent</i>	NARX	S/R	Backpropagation through time	Dynamics approximation, time-series prediction
	HNN	S	Backpropagation through time	Combinatorial optimization, system identification
	LSTM	S/R	Backpropagation through time	Time-series prediction, dynamics approximation, generic function approximation through time



Artificial intelligence and navigation

Introduction to pose estimation

Chapter 9 — Navigation introduced the framework for vision-based relative navigation around both cooperative and uncooperative targets, highlighting the challenges involved in on-orbit servicing and active debris

removal missions [45,46]. Specifically, it was highlighted that if the target satellite is not functional and/or not able to aid the relative navigation (i.e., uncooperative), the estimation of the relative pose by the active servicer spacecraft can represent a critical task. In this context, pose estimation systems based solely on a monocular camera are recently becoming an attractive alternative to systems based on active sensors or stereo cameras, due to their reduced mass, power consumption, and system complexity [47]. However, given the low signal-to-noise ratio (SNR) and the high contrast which characterize space images, a significant effort is still required to comply with most of the demanding requirements for a robust and accurate monocular-based navigation system [48]. Notably, the navigation system cannot rely on known visual markers, as they are typically not installed on an uncooperative target. As already detailed in Chapter 9 – Navigation, the extraction of visual features is an essential step in the pose estimation process, and advanced image processing (IP) techniques are required to extract keypoints (or interest points), corners, and/or edges on the target body. In model-based methods, the detected features are then matched with predefined features on an offline wireframe 3D model of the target to solve for the relative pose. In other words, a reliable detection of key features under adverse orbital conditions is highly desirable to guarantee safe operations around an uncooperative spacecraft.

Despite recent advancements in the field, standard IP techniques generally lack feature detection robustness when applied to space images. From a pose initialization standpoint, the extraction of target features can, in fact, be jeopardized by external factors, such as adverse illumination conditions, low SNR, and the Earth in the background (Fig. 15.29), as well as by target-specific factors, such as the presence of complex textures and features on the target body. At the same time, feature tracking could be complicated by variations of the illumination conditions due to varying Sun-camera geometries, especially when combined with unexpected reflections from different parts of the target body. Moreover, most of IP methods are based on the image gradient, detecting textured-rich features or highly visible parts of the target silhouette. As such, the detected features are image-specific and can vary in number and typology depending on the image histogram. In other words, most of these techniques cannot accommodate an offline feature selection step, which translates into a computationally expensive image-to-model correspondence process to ensure that each detected 2D feature is matched with its 3D counterpart on the available wireframe model of the target object [49].

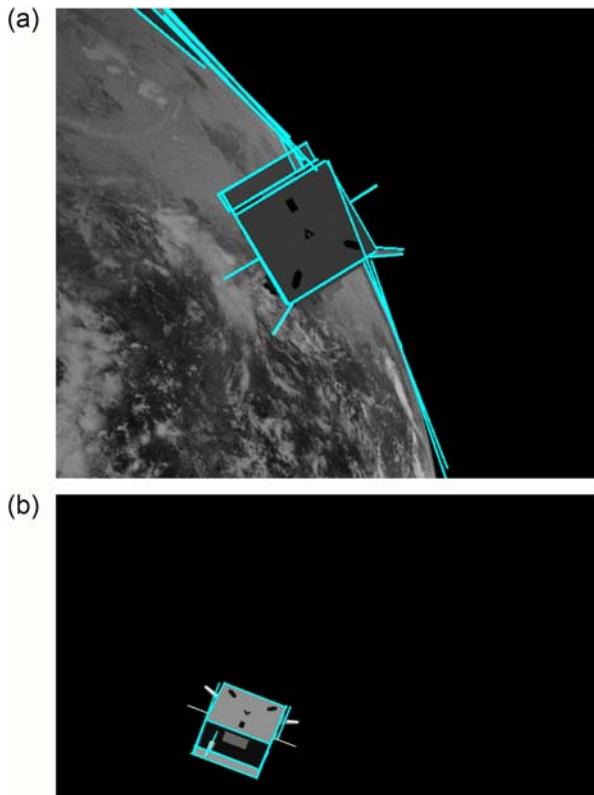


Figure 15.29 Example showing poor detections in images with adverse illumination and background texture using standard IP algorithms. For a correct reference dataset, please refer to Ref. [50].

In this context, AI-based pose estimation architectures are standing out in recent years as a valid and robust alternative to the techniques so far described, mostly because of the strict requirements on the robustness of visual-based relative navigation systems during close-proximity operations around uncooperative targets. Despite most of these methods were already implemented and validated for Earth-based systems, their scalability to space, i.e., their performance in space scenarios, is still under investigation.

AI-based relative pose estimation

Recent advances in CV for pose estimation in terrestrial applications have relied on the quickly evolving domain of ML to enable unprecedented efficacy across several applications. Among all the DL techniques described in

Chapter 15 — Modern Spacecraft GNC — AI in space — Introduction, CNNs stand out as the most viable solution for visual-based systems, mostly due to their capability to process visual data in form of images.

The implementation of CNNs for monocular pose estimation in space has already become an attractive solution in recent years, also thanks to the creation of the Spacecraft PosE Estimation Dataset (SPEED) [51], a database of highly representative synthetic images of PRISMA’s TANGO spacecraft made publicly available by Stanford’s Space Rendezvous Laboratory and applicable to train and test different network architectures.

A high-level schematic of a typical CNN architecture is shown in Fig. 15.30. The first and core building block is represented by the convolutional layer, formed by a fixed number of weight filters which convolve the input image to return feature maps. Generally, a convolution layer uses multiple weight filters on the image. As such, the output is a 3D volume of 2D feature maps. These feature maps are then downsampled by the pooling layer in order to progressively reduce the spatial size of the feature maps, thus reducing the number of parameters and improving the network robustness against overfitting. By cascading convolutional layers with pooling layers in series, the output of the so-called *feature learning* step is a list of feature maps of lower size than the input image. Fully connected layers (i.e., an MLP neural network) are then generally used to classify the image given the input feature maps.

The interesting aspect of CNNs is that the internal representations that express the relationship between two feature maps are automatically generated. As already mentioned, CNNs are capable of extracting abstract representations at unintuitive levels of information in the images to compose high-level information. Therefore, CNNs have been successful in tackling some of the challenges faced by IP systems, where a higher interpretability of the information is required. These include, among others, robustness against:

- viewpoint variation.
- scale variation, deformation, and occlusion.
- illumination conditions.
- background clutter.

Altogether, these are claimed to be the main advantages of CNNs over standard IP algorithms for relative pose estimation [47,52,53]. Notably, CNNs for vision tasks are most often trained offline in a supervised manner, wherein training images are fed to the network and the appropriate convolutional and activation weights are estimated to reduce the loss. This

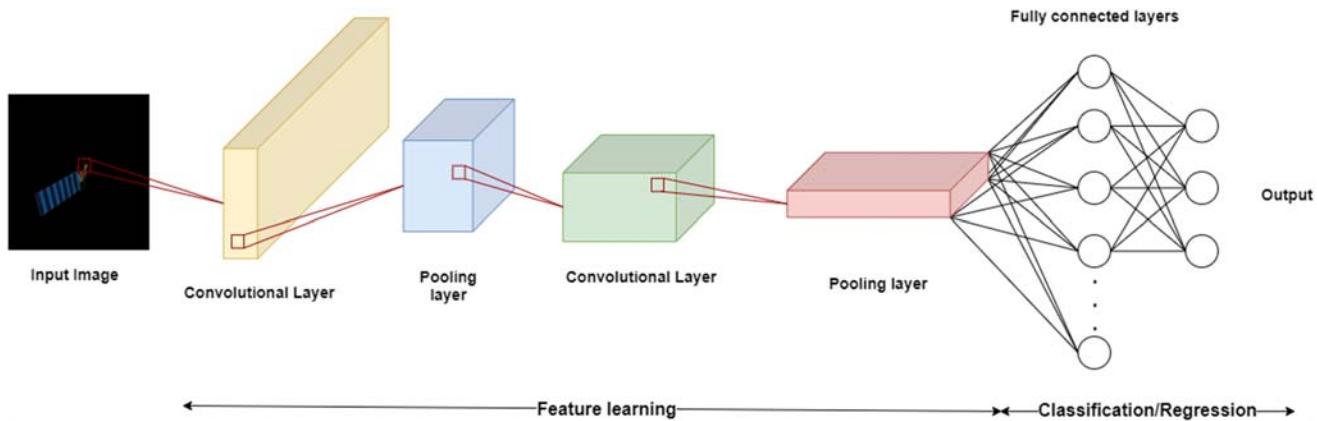


Figure 15.30 High-level schematic of a generic CNN architecture.

guarantees that there is no need to correlate the offline images with the images taken in orbit, as it usually occurs in the feature-matching step of standard keypoints-based methods (cfr. [Chapter 9](#) – Navigation). Nevertheless, CNNs present some drawbacks. First, CNN-based methods require accurate training that typically comprise a wide dataset and high computational power to perform the learning effectively. Moreover, despite the CNN generalization capabilities, the dataset needs to be validated to a certain extent with respect to real images. Finally, dedicated HWs, such as graphics processing units (GPUs) and VPUs (cfr. [Chapter 15](#) – Modern Spacecraft GNC – AI On-board Processors), are typically required to achieve the desired inference time when the application is deployed on-board ([Fig. 15.31](#)).

From a high-level perspective, there are two possible pose estimation systems based on CNNs. These are compared in [Fig. 15.32](#). In each of them, the first essential step is represented by an object detection network (e.g., Faster R-CNN [26], R-FCN [54], or MobileNet-based [55]) placed before the main CNN. The CNNs employed in these detection architectures solve two problems:

1. Type I: regression of bounding box coordinates.
2. Type II: classification of the object class (when more targets could be in the camera FoV).

This is done in order to detect the target object in the image and crop a region of interest (ROI) around it. For monocular pose estimation in space, this step is crucial as ROI cropping allows robustness to scale, variation, and background textures.

In Type I systems, the cropped ROI is fed into a keypoint detection network. These CNNs keep only the feature learning step of [Fig. 15.30](#) and directly output a set of feature maps without classifying the object. These so-called *heatmaps* are detected around preselected features on the



Figure 15.31 Example of heatmaps [48].

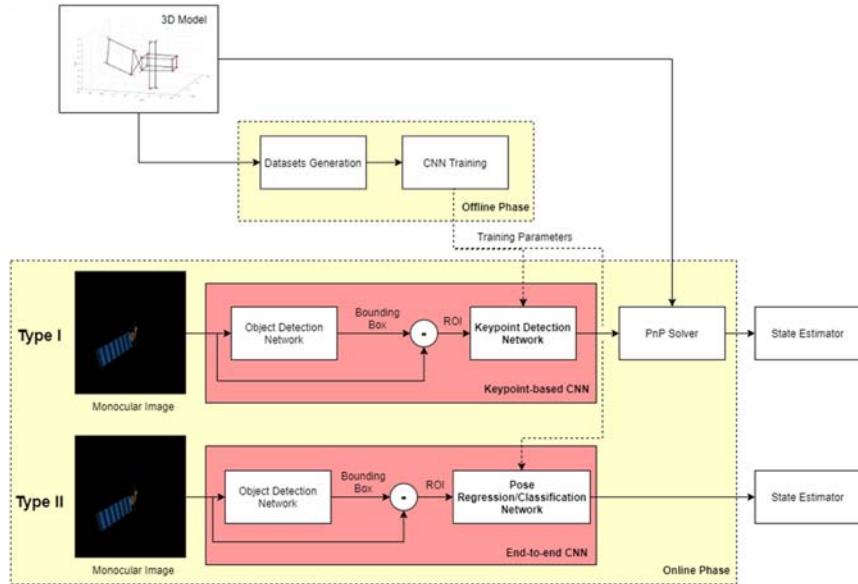


Figure 15.32 Implementation of CNNs for relative pose estimation. Two system architectures are typically used in which the CNN is either used to detect keypoints (Type I) or to directly infer the relative pose (Type II).

target object, such as corners or interest points. The 2D pixel coordinates of the heatmap's peak intensity characterize the predicted feature location, with the intensity and the shape indicating the confidence of locating the corresponding keypoint at this position, i.e., the prediction accuracy.

Notably, the selection of the specific CNN architecture will drive the achievable keypoints detection accuracy and robustness. Some architectures, such as the Hourglass [56] and the U-Net [22], perform a downsampling of the input followed in series by an upsampling, in order to detect features at different scales. However, recent advancements in the field demonstrated that by using parallel subnetworks across multiple resolutions, rather than multiresolution serial stages, the CNN can manage to maintain a richer feature representation, facilitating more accurate and precise heatmaps. For this reason, the HRNet [57] architecture currently represents the state-of-the-art in keypoint detection.

Once the preselected keypoints are detected by the CNN, the extracted keypoints are ultimately fed to a standard perspective-n-point (PnP) solver (cfr. Chapter 9 – Navigation) together with their body coordinates, which are made available through the wireframe 3D model of the target body.

Conversely, Type II systems (also called end-to-end systems) predict the full relative pose vector directly from the input image, based on either classification or regression following the feature learning step (see Fig. 15.30). Classification formulations discretize the pose space and use the fully connected layers to convert the feature maps into a confidence score for each possible pose vector. On the other hand, the regression formulation aims to learn a nonlinear mapping that directly regresses the six pose variables. Pose classification formulation has been utilized in terrestrial applications [58] as well as for uncooperative spacecraft [59], generally using the AlexNet [19] or VGG16 [60] CNN architectures. On the other hand, the regression formulation has been used most often in terrestrial applications [61,62].

Initially, end-to-end CNNs were more commonly exploited for the pose estimation of uncooperative spacecraft [59,63–65]. However, since the pose accuracies of these systems proved to be lower than the accuracies returned by standard PnP solvers, especially in the estimation of the relative attitude, systems based on keypoints-detection recently emerged as the preferred option. Specifically, average orientation errors of $1.31^\circ \pm 2.24^\circ$ were achieved by keypoints-based methods as opposed to the average orientation errors of $9.76^\circ \pm 18.51^\circ$ achieved by end-to-end methods. These averages were computed across test images of the TANGO spacecraft as part of the SPEED challenge [51].

Notably, another advantage of keypoints-based methods resides in their capability to return both the 2D location of the detected features (CNN output) and the full relative pose (PnP output). Consequently, they can guarantee a much more flexible interface with the navigation filter, as discussed in the next section.

Interface with navigation and uncertainty estimate

The AI-based methods discussed in the previous section can provide an estimate of the relative pose for a static monocular image. This initial estimate of the relative pose of the target spacecraft with respect to the servicer spacecraft does not require a priori information of the relative state and can therefore be exploited in lost-in-space scenarios. Due to their computationally expensive IP, these pose initialization routines are, however, not well suited to produce estimates at the high frequencies usually required during close proximity operations. Therefore, a navigation filter is typically used in combination with the camera measurements and the pose estimates in order to return relative state solutions at high frequencies [66]. The internal dynamics

of the filter can improve the accuracy of the predicted relative state from measurements and allow a more robust pose tracking while estimating the relative translational and rotational velocities.

As already discussed in the previous section, one of the major differences between end-to-end and keypoints-based pose estimation systems resides in the intermediate estimates that can be output at each estimation step. In a keypoints-based system (Type I), both keypoints location and relative pose can be estimated by cascading an AI-based keypoints detector with a standard PnP solver. Conversely, only an estimate of the relative pose can be output by an end-to-end system (Type II). As a result, the selection of the pose estimation system will drive the selection of the type of navigation filter.

From a high-level perspective, two different navigation architectures are normally exploited in the framework of relative pose estimation (Fig. 15.33). As discussed in Chapter 9 – Navigation, a *tightly coupled* architecture, where the features are directly processed by the navigation filter as measurements, and a *loosely coupled* architecture, in which the relative pose is directly fed to the filter as pseudo-measurement. For each of these types, several navigation filters can be adopted, cfr. Chapter 9 – Navigation. The reader is referred to Ref. [48] for a more comprehensive overview. Remarkably, as detailed in Chapter 9 – Navigation, recent implementations adopted a multiplicative extended Kalman filter (MEKF) based on modified Rodrigues parameters to overcome the covariance instabilities resulting from using quaternions as attitude representation [67]. A three-element error parametrization a ,

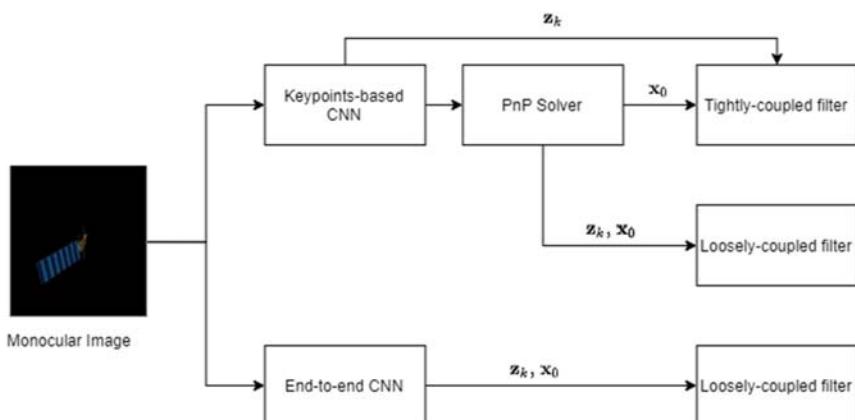


Figure 15.33 Overview of the three different interfaces between the CNN, the PnP solver, and the navigation filter.

expressed in terms of quaternions, is propagated and corrected inside the filter to return an estimate of the attitude error. At each estimation step, this error estimate is used to update a reference quaternion and is reset to zero for the next iteration. Notably, the reset step prevents the attitude error parametrization from reaching singularities, which generally occur for large angles. Assuming a typical state vector for relative pose estimation:

$$\mathbf{x} = \begin{pmatrix} \mathbf{r} \\ \mathbf{v} \\ \mathbf{q} \\ \boldsymbol{\omega} \end{pmatrix}$$

The propagation steps follow the derivation described in [Chapter 4](#) – Orbital Dynamics, [Chapter 5](#) – Attitude Dynamics and [Chapter 9](#) – Navigation. In an MEKF, the correction steps are identical to the standard Kalman filter formulation (cfr. [Chapter 9](#) – Navigation):

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - h(\hat{\mathbf{x}}_k^-, 0))$$

where $\hat{\mathbf{x}}_k^-$ represents the state propagated by the filter internal dynamics, \mathbf{K}_k is the Kalman gain, \mathbf{y}_k is the measurements vector, and $h(\hat{\mathbf{x}}_k^-, 0)$ is the nonlinear model of the measurements as a function of the propagated state. As described in [Chapter 9](#) – Navigation, in a loosely coupled filter, the measurements are directly represented by the relative pose between the servicer and the target spacecraft. In this case, a pseudo-measurements vector is derived by transforming the relative quaternion set into the desired attitude error \mathbf{a} that represents a useful variable to work with:

$$\delta \mathbf{q}_y = \mathbf{q}_y \times \hat{\mathbf{q}}_k^{-*} \rightarrow \mathbf{a} = 4 \frac{\delta \mathbf{q}_{13}}{1 + \delta q_4}$$

where \times denotes the quaternion product, \mathbf{q}_y is the quaternion measurement, and $\hat{\mathbf{q}}_k^{-*}$ represents the estimated quaternion conjugate at the previous step, namely $\hat{\mathbf{q}}_k^-$. The measurements update equation then becomes simply:

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k = \begin{bmatrix} I_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & I_3 & 0_3 \end{bmatrix} \hat{\mathbf{x}}_k^- + \mathbf{v}_k$$

where the Jacobian \mathbf{H}_k represents the linearization of the measurements model $h(\hat{\mathbf{x}}_k^-, 0)$. Conversely, in a tightly coupled filter, the measurements are represented by the pixel coordinates of the detected features, and the measurements update equation can be written as:

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k = [H_{r,i} \mathbf{0}_{2n \times 3} H_{a,i} \mathbf{0}_{2n \times 3} \vdots \vdots \vdots H_{r,n} \mathbf{0}_{2n \times 3} H_{a,n} \mathbf{0}_{2n \times 3}] \hat{\mathbf{x}}_k^- + \mathbf{v}_k$$

where the partial derivatives of the $2n$ pixel coordinates of the detected n features have to be computed with respect to the filter state, significantly increasing the computational burden.

In the tightly coupled filter, the measurement covariance matrix \mathbf{R} is a block diagonal matrix constructed with the covariances of each detected feature:

$$\mathbf{R} = [\mathbf{C}_1 \ddots \mathbf{C}_n]$$

If a keypoints regression architecture (Type I) is chosen at a pose estimation level, a covariance matrix can be derived for each detected feature, taking advantage of their shape by associating small and large covariances to highly symmetrical and largely asymmetrical heatmaps, respectively. Consequently, a more representative statistical information can be associated to the filter measurements and improve the overall filter robustness.

Notably, the prediction accuracy returned by the CNN for each heatmap can also be used to discard outliers, allowing a more reliable set of measurements handled by the navigation filter.

Conversely, in the loosely coupled filter, the measurements matrix \mathbf{R} represents the uncertainty in the pose estimation step, and hence it cannot be directly related to the CNN heatmaps. In case a keypoints-based CNN is exploited together with a standard PnP solver (Type I), no statistical information can be easily retrieved from the pose estimation, and hence a constant covariance is usually chosen based on the pose estimation accuracy observed for the validation or test datasets. If on the other hand an end-to-end CNN is used (Type II), the single CNN confidence returned for each estimated pose cannot be easily associated to each element of the relative position and attitude. To compensate for this lack of statistical information, recent implementations [68] accommodated an error prediction network after the main end-to-end CNN which rejects wrong pose estimates. As such, inaccurate measurements can be rejected prior to the navigation step and a constant measurements matrix \mathbf{R} can be used without undermining filter robustness.

Use case scenario: keypoints regression architecture

An important application of AI-based pose estimation systems for close-proximity operations around a target is when the target is uncooperative and tumbling, requiring a highly accurate and robust pose estimation system. From a navigation filter perspective, a loosely coupled approach is usually preferred for a tumbling target, due to the fact that the fast relative dynamics could jeopardize feature tracking and return highly variable measurements to the filter. However, the fact that it is generally hard to obtain a representative covariance matrix for the pseudo-measurements becomes critical when filter robustness is demanded. Remarkably, the adoption of a keypoints-based CNN system can overcome the challenges in feature tracking by guaranteeing the detection of a constant, predefined set of features. Moreover, the CNN heatmaps can be used to derive a measurements covariance matrix and improve filter robustness, see Fig. 15.34.

In this context, this section describes the main steps of a keypoints-based CNN pose estimation system required for the pose and features initialization of the navigation filter prior to tracking [69]. The European Space Agency's (ESA's) Envisat spacecraft is selected as a use case scenario, as Envisat has been identified as the single most risk-inducing piece of space debris in

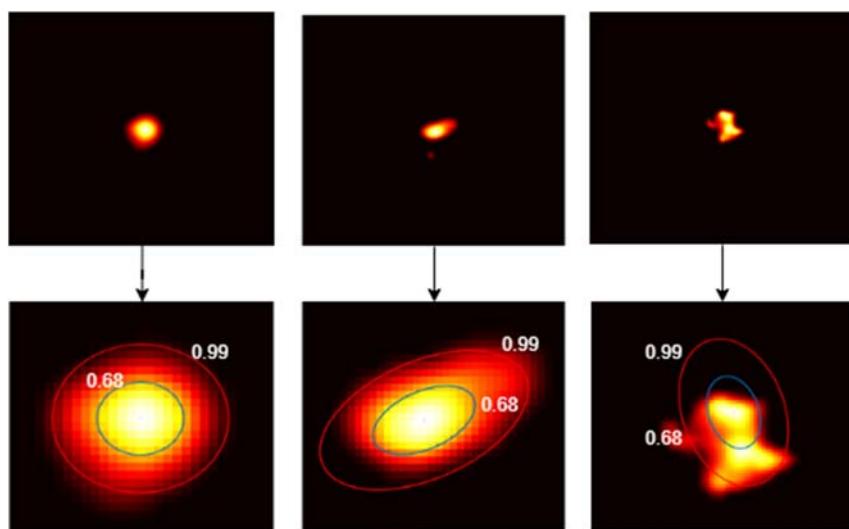


Figure 15.34 Heatmaps illustration for an accurate (left), slightly inaccurate (center), and inaccurate (right) detection of a typical CNN. The displayed ellipses are derived from the computed covariances by assuming the confidence intervals $1\sigma = 0.68$ and $1\sigma = 0.99$.

multiple independent studies [70,71]. Fig. 15.35 illustrates the main steps together with the intermediate outputs of each subsystem. The corners of the Envisat body and of the specific absorption rate antenna are selected as trainable features for the CNN as well as 3D points in the PnP solver. These keypoints are chosen since they are more invariant to surface degradation than richer features on the target body. Notice that since the tasks associated to the object detection network go beyond the scope of this section, the following description assumes that an ROI has already been cropped around the target spacecraft.

Keypoints detection network – training, testing, and inference

The selected keypoints-based HRNet is trained offline in a supervised way on ~ 10.000 synthetic images of Envisat. Ideally, illumination conditions as well as camera views and image noise should be optimized in order to guarantee a complete and representative dataset of the tumbling target. During training, the validation dataset is used beside the training one to compute the validation losses and avoid overfitting. The network is trained for a total of 210 epochs with the Adam optimizer [73]. A decaying learning rate with initial value of 10^{-3} and decaying factor of 0.1 is used to optimize the final tuning of the network's weights. Finally, the network performance after training can be assessed with the test dataset. A ratio of $\sim 70\%$, $\sim 15\%$, and $\sim 15\%$ is usually chosen for training, validation, and testing, respectively. After the offline phase, the CNN can be used to perform keypoints inference on previously unseen images of Envisat. Fig. 15.36 shows two of the heatmaps detected by the HRNet architecture for a sample monocular image of Envisat, whereas Fig. 15.37 shows the pixel location of the 12 keypoints, extracted from each heatmap by computing the coordinates of their peak intensity. Notably, accurate detections are characterized by heatmaps in which the peak location can be easily computed from the R channel of the RGB output image. For the selected monocular image of Envisat, the mean detection accuracy across all the 12 keypoints amounts to 0.67 pixels, with a maximum detection error of 1.6 pixels. Furthermore, one key advantage of relying on CNNs for feature detection can be found in the capability of learning the relative position between features which are not visible due to adverse illumination and/or occulted by other parts of the target.

Covariance computation

To derive a covariance matrix for each feature directly from the heatmaps detected by the CNN, the first step is to obtain a statistical population

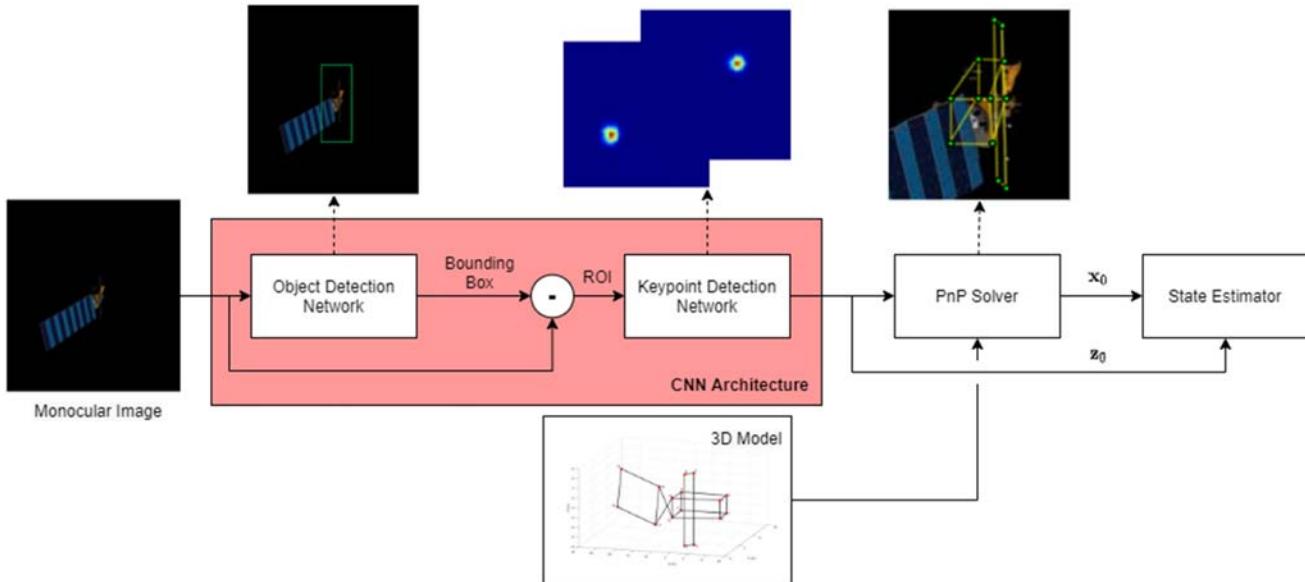


Figure 15.35 Overview of the keypoints-based CNN system and its interface with the state estimator. Adapted from L. Pasqualetto Cassinis, A. Menicucci, E. Gill, I. Ahrns, M. Sanchez-Gestido, *On-ground validation of a CNN-based monocular pose estimation system for uncooperative spacecraft: bridging domain shift in rendezvous scenarios*, *Acta Astronautica*, 196 (2022) 123–138. <https://doi.org/10.1016/j.actaastro.2022.04.002>.

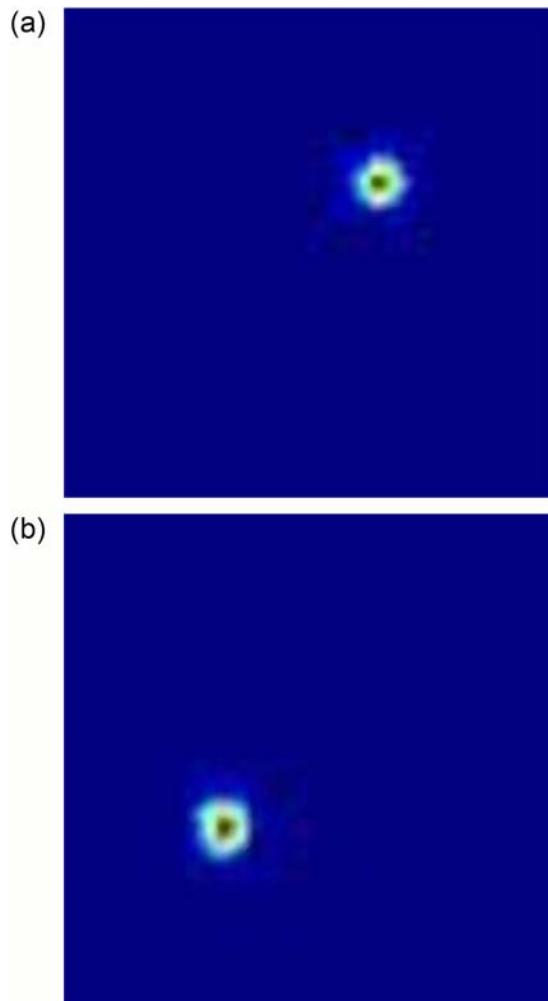


Figure 15.36 Example of heatmaps output for two accurate HRNet detections.

around the heatmap's peak. This is done by thresholding each heatmap image so that only the x - and y -locations of heatmap's pixels are extracted. Secondly, each pixel within the population is given a normalized weight based on the gray intensity at its location. This is done in order to give more weight to pixels which are particularly bright and close to the peak, and less weight to pixels which are very faint and far from the peak. Finally, the obtained statistical population of each feature is used to compute the weighted covariance between x , y and consequently the covariance matrix

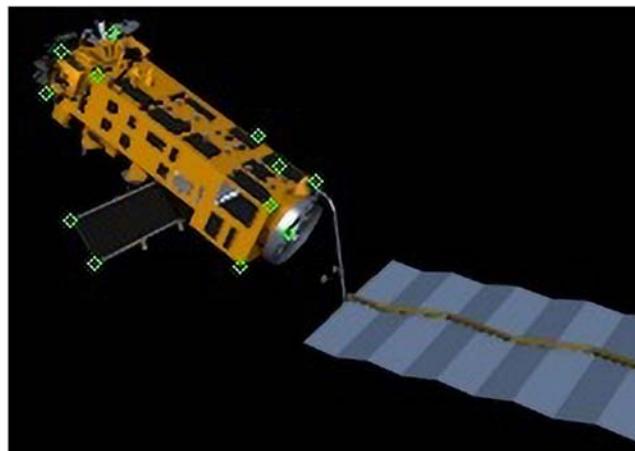


Figure 15.37 Pixel location of the 12 detected Envisat keypoints. As it can be seen, partially occluded corners are still detected by the CNN, thanks to its capability to infer the geometrical relation between each corner.

\mathbf{C}_i for each feature. As all the detected heatmaps represent accurate detections, their associated covariances are nearly diagonal and characterized by diagonal elements of $\sim 1\text{--}2$ pixels. For the first and last features, this corresponds to the following covariance matrices:

$$\mathbf{C}_1 = [1.97 \ 0.14 \ 0.14 \ 2.11], \mathbf{C}_{12} = [1.65 \ 0.02 \ 0.02 \ 1.56]$$

PnP solver for state filter initialization

The pixel coordinates of the 12 detected features are fed into a standard PnP solver together with their 3D body coordinates and the camera intrinsic parameters to solve for the initial relative pose between the monocular camera and Envisat. Due to the high pixel accuracies obtained in the CNN-based features detection step, accurate results are achieved at pose estimation level. Specifically, an error of 0.3 m and 0.5 degrees is obtained for the norm of the relative position and for the Euler axis-angle, respectively.

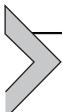
State estimator

The relative pose estimated by the PnP solver, together with assumed values for the translational and rotational velocities, is used as initial guess x_0 for the navigation filter. The estimated x, y pixel coordinates of the 12 preselected features are instead concatenated into a $2n$ measurements column vector

z. Likewise, the feature covariances \mathbf{C}_i are concatenated into the block-diagonal measurement matrix \mathbf{R} :

$$\mathbf{R} = [(1.97 \ 0.14 \ 0.14 \ 2.11) \ddots (1.65 \ 0.02 \ 0.02 \ 1.56)]$$

After initialization, the filter estimates the full relative state by propagating the relative pose and correcting it with the measured features. Notice that the single covariance can differ for each feature at a given time step as well as vary over time. As a result, a heatmaps-derived covariance matrix can capture the statistical distribution of the measured features and improve the measurements update step of the navigation filter.



Validation of AI-based systems

In the previous sections of Chapter 15—Modern GNC, the validation of AI-based pose estimation architectures was introduced as an offline–online process in which synthetic datasets, representative of a 3D model of the target, were used to train, validate, and test the CNNs (offline phase) prior to the inference of the relative pose on a sequence of images (online phase). Overall, such *synthetic validation* consists in assessing the pose estimation accuracy of the AI-based system on synthetic images that were previously unseen by the CNN in the offline phase. However, a more representative on-ground validation of the CNNs’ performance shall be sought to test the system robustness against representative images of the target spacecraft. Due to a lack of large datasets of actual space imagery, these images are usually generated in laboratory environments which recreate space-like illumination conditions.

At the time of writing, several laboratory setups exist to recreate rendezvous approaches around a mockup of a target space object (i.e., asteroid, moon surface, or spacecraft) with a monocular camera [74], i.e., the Testbed for Rendezvous and Optical Navigation (TRON) at Stanford University [51], the GNC Rendezvous, Approach and Landing Simulator (GRALS) at the European Space Research and Technology Centre (ESTEC) [75], the European Proximity Operations Simulator (EPOS) at the German Aerospace Agency (DLR) [76], the Platform-art facility at GMV [77], and the ARGOS facility at Politecnico di Milano [78,79]. In these setups, the on-ground validation is achieved by establishing a calibration framework to associate to each generated image a representative estimate (GT) of the relative pose between the adopted monocular camera and a mock-up of

the target. This GT pose is then compared to the AI-based estimate to assess the accuracy and robustness of the pose estimation architecture.

In this context, a fundamental challenge arises from the need to bridge the quality gap between the synthetic renderings and the lab-representative images. If a synthetic dataset used to train the adopted CNN fails in representing the textures of the target mockup as well as the specific illumination in the laboratory setup, the performance on lab-generated images will, in fact, result in inaccurate pose estimates. To overcome this, recent works addressed the impact of augmented synthetic datasets on the CNN performance in either lab-generated or space-based imagery [15,50,68,69,72,80]. These augmented datasets are built on a backbone of purely synthetic images of the target by adding noise, randomized and real Earth background, and randomized textures of the target model.

CNN validation – methods and metrics

In relation to Fig. 15.38, the on-ground validation can be seen as a result of several intermediate procedures aimed at establishing a comparative framework between the estimated and true quantities.

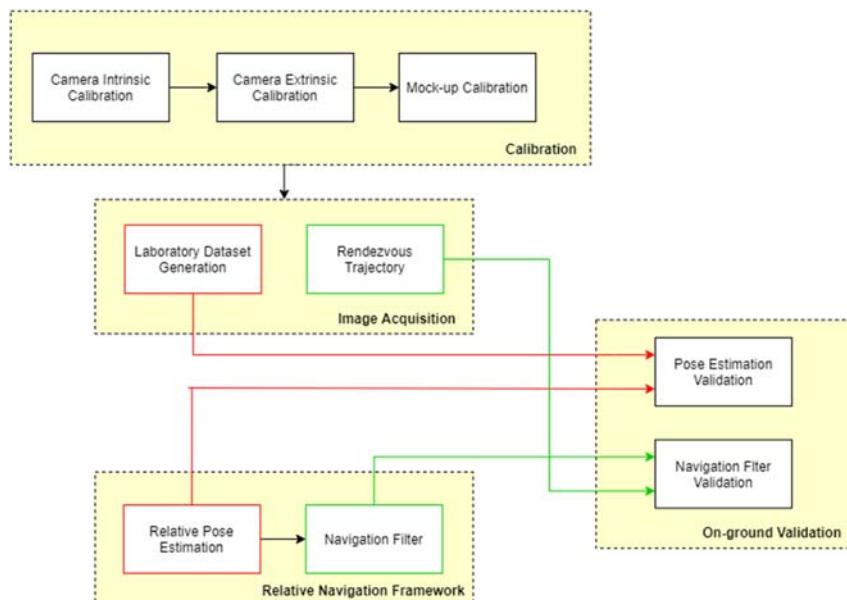


Figure 15.38 High-level overview of the on-ground validation of the relative navigation framework.

The first step usually seen in the current setups consists in the calibration of the laboratory environment. In open-loop tests, this is done in order to be able to accurately estimate the true relative pose between the monocular camera and the target mock-up. In closed-loop tests, however, the calibration of the camera with respect to its robotic arm (hand-eye calibration) is additionally required. This is crucial if the motion of the robotic arm shall be determined which moves the camera at the desired relative pose from the target mock-up.

Once the setup is calibrated, the actual image acquisition procedures can be executed, and each generated image can be associated to its true relative pose. A laboratory dataset of static images as well as a continuous rendezvous trajectory sequence can be recreated, depending on the main objective of the validation. Typically, datasets are recreated first to validate the CNN performance at a pose estimation level. Once the CNN accuracy is assessed, the full navigation framework can then be validated in open/closed loop against the representative images of a rendezvous trajectory.

Camera intrinsic calibration

The first step of the calibration procedure consists of the estimation of the camera intrinsic parameters, such as the focal length, the principal point, and the tangential and radial distortion coefficients. This is generally accomplished by taking images of known markers, such as Haruko-based chessboard or traditional chessboard, with different camera views and solving a standard PnP problem, in which the intrinsic camera parameters and the relative poses between each camera view and the chessboard are the unknowns.

Camera-to-mockup extrinsic calibration

The extrinsic calibration consists of accurately estimating the relative pose between the adopted monocular camera and the mockup of the target object. This is achieved through the calibration of each object with respect to a fixed reference frame within the laboratory environment.

From a high-level perspective, most of the current calibration procedures carried out prior to open-loop tests can be described through Fig. 15.39, which represents the GRALS calibration setup adopted in Ref. [72]. The setups are constituted of the following elements: (a) a scaled mockup of the target object; (b) a monocular camera mounted on a robotic arm, and (c) a tracking system used to track objects with retroreflective/IR markers and to provide estimates of their pose with respect to a user-defined reference frame.

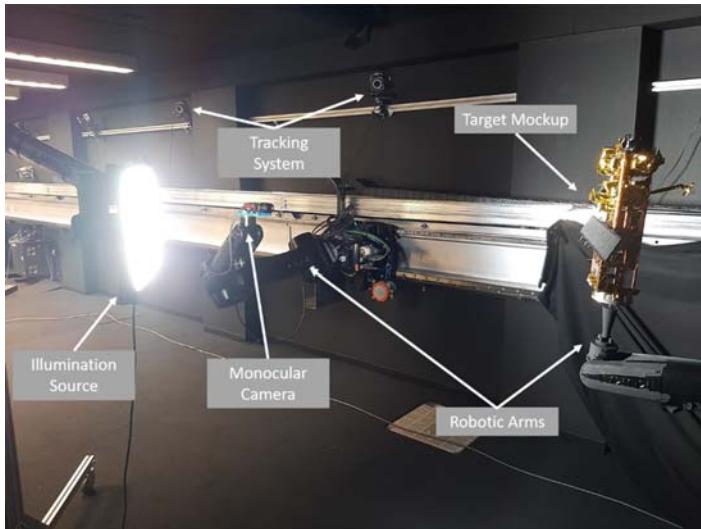


Figure 15.39 Representation of the ESTEC GRALS calibration setup followed prior to the image acquisition step [72].

The mockup of the Envisat spacecraft is calibrated with respect to a monocular camera mounted on a KUKA robotic arm. The reference GT of the relative pose is estimated by calibrating both objects with respect to the fixed reference frame of the VICON tracking system, which is used to track objects within the laboratory environment through a set of IR cameras. The procedure usually consists of the following steps:

- Camera-to-laboratory calibration, in which the camera is extrinsically calibrated with respect to a fixed reference frame, usually the tracker frame or the robot frame (VICON tracking system and KUKA robots in Fig. 15.39).
- Mockup-to-laboratory calibration, in which the target mockup is calibrated with respect to the same fixed reference frames.
- Camera-to-mockup calibration, in which the relative pose between the two objects can be obtained through the selected fixed reference frame.

Closed-loop hand-eye calibration

Several hand-eye algorithms exist in literature to calibrate a camera with respect to a generic robotic arm [81], each of them handling the problem in different ways. However, the two inputs to all of them consists of:

- Camera extrinsic parameters for each chessboard image, i.e., relative pose of the camera with respect to the chessboard frame from the intrinsic calibration step.

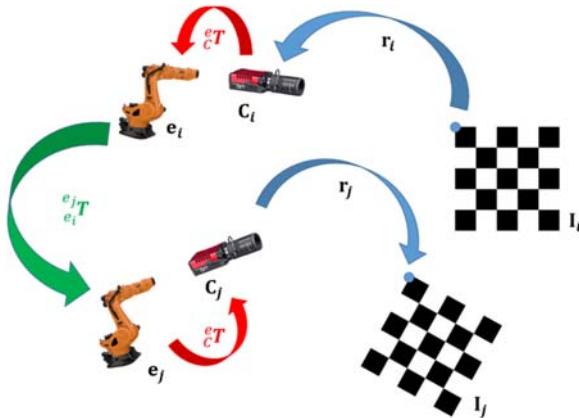


Figure 15.40 Transformation required to reproject a point from image i to image $j = i + 1$ through the hand-eye calibration. The transformation labeled in red represents the solution of the hand-eye calibration, whereas the transformation labeled in green expresses the relative pose between two different end effector configurations.

- Telemetry recordings of the end effector pose with respect to the robot frame for each of the chessboard images.

Once the hand-eye transformation is estimated, the accuracy of the estimate can be assessed by computing the reprojection error of each chessboard corners from image i to image $j = i + 1$. Referring to Fig. 15.40, this is achieved by first expressing the 3D location of each point $\mathbf{r}_{k,i}$ in the camera frame when the camera is in the i -th location. Then, by using the estimated hand-eye transformation \mathbf{CeT} and the transformation $\mathbf{e}_i \mathbf{e}_j \mathbf{T}$ to go from the end effector \mathbf{e}_i to the end effector \mathbf{e}_j , the 3D location $\mathbf{r}_{k,j}$ in the camera frame when the camera is in the j -th location can be found and, consequently, the point reprojection in image j :

$$\mathbf{r}_j = (\mathbf{CeT})^{-1} \mathbf{e}_i \mathbf{e}_j \mathbf{T} \mathbf{CeT}^{-1} \mathbf{r}_i$$

Error metrics

In case a keypoints-based CNN is used (Type 1), the keypoints detection performance can be assessed in terms of root mean squared error between the GT and the x , y coordinates of the extracted features, which is computed as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n_{tot}} \left[(x_{GT,i} - x_i)^2 + (y_{GT,i} - y_i)^2 \right]}{n_{tot}}}$$

Conversely, two separate error metrics are adopted in the evaluation of the pose estimation performance [51]. Firstly, the translational error between the estimated relative position $\hat{\mathbf{t}}^C$ and the GT \mathbf{t}_{GT}^C is computed as:

$$E_T = \|\mathbf{t}_{GT}^C - \hat{\mathbf{t}}^C\|$$

This metric is also applied for the translational and rotational velocities estimated in the navigation filter. Secondly, the attitude accuracy is measured in terms of the Euler axis-angle error between the estimated quaternion $\hat{\mathbf{q}}$ and the GT \mathbf{q}_{GT} :

$$\boldsymbol{\beta} = (\boldsymbol{\beta}_s \boldsymbol{\beta}_v) = \mathbf{q}_{GT} \times \hat{\mathbf{q}}$$

$$E_R = (|\boldsymbol{\beta}_s|)$$

Training augmentation techniques

Referring to Fig. 15.41, the first step of the currently adopted pipelines for the datasets augmentation and randomization [68,72,80] consists in generating ideal synthetic images of a CAD model of the target. Generally, the Azimuth and Elevation of the Sun are randomly varied around the ideal camera-Sun relative position, in order to recreate favorable as well as more adverse illumination conditions. Next, a randomization pipeline is introduced which adds the following effects to the rendering:

- *Texture randomization.* This is performed in order to increase the CNN robustness against texture variations between the synthetic and laboratory mockup of the target.
- *Light randomization.* Additional lights are introduced in random locations, aside from the main Sun illumination, in order to increase the CNN robustness against the illumination conditions recreated in the laboratory setup
- *Background randomization.* Random scenes are used as image background in order to increase the CNN robustness against the laboratory environment. Specifically, external disturbance sources in the lab are likely to return nonzero pixel values in the image background, leading to inaccurate CNN detections in case the training dataset would consist of only black backgrounds.

Following the software (SW) rendering, an additional pipeline is used to further augment the generated images. This is performed by introducing the Earth in the background in some of the images and by corrupting the images with the following noise models:

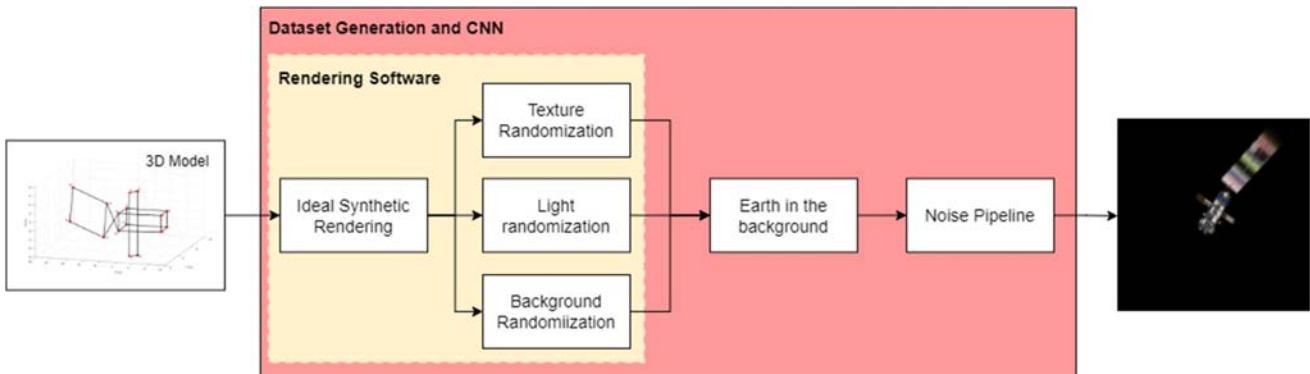


Figure 15.41 High-level schematic of an image augmentation pipeline. Adapted from L. Pasqualetto Cassinis, A. Menicucci, E. Gill, I. Ahrns, M. Sanchez-Gestido, On-ground validation of a CNN-based monocular pose estimation system for uncooperative spacecraft: bridging domain shift in rendezvous scenarios, *Acta Astronautica*, 196 (2022) 123–138. <https://doi.org/10.1016/j.actaastro.2022.04.002>.

- Gaussian, shot, impulse, and speckle noise.
- Gaussian, defocus, motion, and zoom blurs.
- Spatter, color jitter, and random erase.

Use case scenario — validation of keypoints detection accuracy

In Type-I pose estimation systems, the system performance on the laboratory dataset can be evaluated at both keypoints detection and pose estimation levels. To show the importance of dataset randomization in the training phase, the keypoints detection network, trained with the randomized training dataset, is compared with the keypoints detection of the same CNN trained on a subset of the augmented dataset, characterized only by the Earth in the background and noise. This is shown in Fig. 15.42 for a sample image. Due to a lack of background, light, and texture randomization in the training dataset, the CNN trained only on the partially augmented dataset is overfitted on the textures learned on the synthetic Envisat model. As a result, the network cannot associate the correct texture to each feature, and the detected keypoints are randomly scattered around the image (Fig. 15.42A). Conversely, the CNN trained on the randomized dataset proves to be more robust against variations in texture and light between the synthetic and the lab images, inferring the correct shape of the mockup and detecting most of the keypoints in the correct location (Fig. 15.42B). This improved robustness is mostly linked to the capability of the CNN to learn shapes rather than textures, which can be traced back to the textures' randomization step included in the augmentation and randomization pipeline. Remarkably, the features are detected even without a high synthetic-lab representativeness, showing the CNN capability to transfer to images which considerably differ from the training ones.

Reinforcement learning

The classification for ML approaches is composed by three categories that depend on the nature of the feedback available to the learning system. They are supervised learning, unsupervised learning, and reinforcement learning, as already discussed previously.

Nowadays research in the field of ML is focused especially on supervised learning. It exploits a training set of examples provided by an external supervisor; all these examples are description of a situation in a specific environment, intended as the correct action that the system must take when it finds

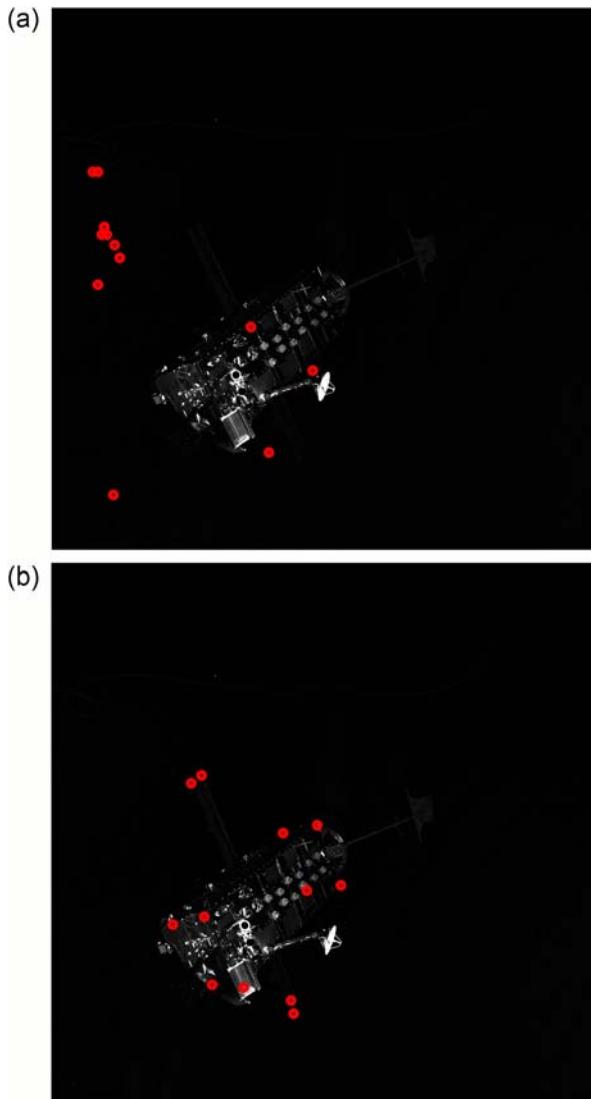


Figure 15.42 Impact of light, textures, and background randomization on the CNN detection performance for a sample laboratory image of the Envisat mockup [72]. Notably, the randomization of the training dataset improves the CNN robustness against different light, texture, and background conditions.

itself in a particular state. The final objective of training an agent with supervised learning is the achievement of the agent awareness, thanks to whom the agent is capable to act correctly also in situations that are not present in the training set. This is a very powerful way to solve problems like image recognition, but it is not so adequate for learning from interaction. Indeed,

in interactive problems, it is almost always impossible to obtain a training set of the desired behavior for all the situations in which the agent could act. In these kinds of problems, the best way for the agent to learn is doing from its own experience. The methodology called unsupervised learning is aimed to find structures hidden in collections of unlabeled data. Therefore, if for supervised learning the agent is fed by example inputs and desired outputs, in unsupervised learning, the agent is left on its own to find the structure in its inputs because no a priori knowledge is presented as outputs.

Apart from these two methods that seem to exhaustively classify the ML paradigms, the reinforcement learning is the last component of this picture. Reinforcement learning is an AI paradigm that consists in learning what to do, how to map situations to actions, to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them [82]. A founding concept of reinforcement learning is the Markov decision process (MDP), summarized below.

MDP: The MDP is a discrete-time stochastic control process that does not possess any memory. It was theorized by Bellman in the '1950s by extending the concept of Markov chains, proposed by the Russian mathematician Markov. The MDP presents the problem of decision control as a sequential decision process in which, at each time instant, when the process is in a given state, the decision maker takes an action that moves the system to a new state. A particular finite MDP is defined by its state and action sets and by the one-step dynamics of the environment. The transition probability between the two states depends only on the particular action-state pair and not on past states, because of the absence of memory, and can be associated to a scalar reward that establishes the goodness of the action taken in that particular state.

Differently from unsupervised learning, it tries to maximize a reward instead of finding hidden structures from a set of given data. Therefore, reinforcement learning can be defined as a tool aimed to learn a policy in order to maximize a numerical reward. The two important characteristics of this method are the trial-and-error search and the reward, and both of them distinguish it from the other methods. In fact, the agent, or learner, of the problem at the beginning does not know which actions to take, indeed its task becomes to discover which are the actions that yield the greatest reward value. The actions taken may affect not only the immediate reward but also the future development of the problem and then all the sequence of rewards. Reinforcement learning is one of the most common methods used to solve

MDP problems both fully and partially observable. The final goal is to find a policy capable to select the right action to take when the agent is in a particular state. Policies may also be found with evolutionary methods, but they are capable to achieve a feasible solution only when the search space is small, otherwise the optimization becomes computationally intractable. MDPs have three aspects: sensation, action, and goal. Indeed, the agent is called to sense the state in its environment and therefore to take an action that affects the state itself, to achieve the goal or goals of the agent in that specific environment. One of the most challenging features of reinforcement learning that differentiates it from the others kind of learning is the trade-off between exploration and exploitation. Logically, to obtain a good reward, an agent should prefer actions already tried in the past that have been found to be effective for the final reward value. But, on the contrary, to find such kind of actions, the same agent should explore actions that it has never selected before. Therefore, at each step, the agent has the dilemma of either exploiting already experienced situations with sure results or to explore new situations to find better actions for the future. The exploitation-exploration trade-off, then, must be considered as an agent that wants to try a variety of actions and progressively favor the ones that appear to work better.

The agent typically needs to explore the environment in order to learn a proper optimal policy, which determines the required action at a given perceived state. At the same time, the agent needs to exploit such information to actually carry out the task. In the space domain, and especially in on-line applications, the balance must be shifted toward exploitation, for practical reasons. Another distinction that ought to be made is between model-free and model-based reinforcement learning techniques, as shown in [Table 15.2](#). Model-based methods rely on planning as their primary component, while model-free methods primarily rely on learning. Although

Table 15.2 Model-based versus model-free reinforcement learning.

Model-free	Model-based
Unknown system dynamics	Learnt system dynamics
The agent is not able to make predictions	The agent exploits predictive planning
Extensive need for exploration	Minimal need for exploration
Lower computational cost while executing	Higher computational cost while executing

there are real differences between these two kinds of methods, there are also great similarities. We call environmental model whatever information the agent can use to make predictions on what will be the reaction of the environment to a certain action. For the reasons above, the model-based approach seems to be beneficial in the context of spacecraft guidance and control, as it merges the advantage of analytical base models, learning, and planning.

It is important to report some of the key concepts of reinforcement learning:

- *Policy.* It defines the learning agent's way of behaving at a given time. Practically, it can be considered as a map from the states to the actions that can be taken in an environment. It can be a simple function or a very complex element that requires extensive computations. In general, a policy is a stochastic element that specifies the probability to take an action when the agent is in a defined state.
- *Reward.* At each time step, the environment sends to the reinforcement learning agent a scalar number called the reward. The agent's objective is to maximize the total reward, and thus the reward function is a mathematical representation of what is good or bad for the agent in an immediate sense; it is the primary reason based on which the policy changes over time. In general, the reward function depends on the state of the environment and on the actions taken.
- *Value function.* The value function determines what is good in the long run. It represents the total amount of reward that an agent can expect to accumulate in the future, starting from the state in which it is computed. Whereas rewards are the immediate desirability of the state, the values are the long-term desirability of states after considering the states that are likely to follow and the rewards achievable in those states, i.e., a state may yield a low reward but still have a high value because it is followed by other states that bring higher rewards.
- *Model.* The model of the environment is the element that simulates the real behavior of the environment. It is used as a planning method by which an action can be decided, considering possible future situations before the agent experiences them. Reinforcement learning methods that use prebuilt environment models are called model-based methods, otherwise they are model-free methods.

The standard reinforcement learning theory states that an agent can obtain a policy, which provides the mapping between a state $x \in X$, where X is the set of possible states, to an action $a \in A$ where A is the set of possible

actions. The dynamics of the agent is basically represented by a transition probability $p(x_{k+1}|x_k, a_k)$ from one state to another at a given time step. In general, the learnt policy can be deterministic $\pi(x_k)$ or stochastic $\pi(a_k|x_k)$, meaning that the control action follows a conditional probability distribution across the states. Every time the agent performs an action, it receives a reward $r(x_k, a_k)$: the ultimate goal of the agent is to maximize the

accumulated discounted reward $R = \sum_{i=k}^N \gamma^{i-k} r(x_i, a_i)$ from a given time

step k to the end of the horizon N which could be $N = \infty$ in infinite horizon. The user-defined coefficient γ is the tunable discount parameter to define how the agent pursues future rewards. As mentioned, the value function V^π is the total amount of reward an agent can expect to accumulate in the future, at a given state. Note that the value function is obviously associated to a policy:

$$V^{(\pi(x_k))} = E[R|x_k, a_k = \pi(x_k)]$$

In most of the reinforcement learning applications, a very important concept is the action-value function Q^π :

$$Q^\pi(x_k, a_k) = r(x_k, a_k) + \gamma \sum_{x_{k+1}} p(x_{k+1}|x_k, a_k) V^\pi(x_{k+1})$$

in which the remarkable difference with the value function is the fact that the action-value function refers to the expected cumulative reward at a certain state, given a certain action. The optimal policy is the one that maximizes the value function $\tilde{\pi} = \text{argmax}_\pi V^\pi(x_k)$.

The reinforcement learning problem has been tackled using several approaches, which can be divided into two main categories: the policy-based methods and the value-based methods. The former ones search for the policy that behaves correctly in a specific environment; the latter ones try to value the utility of taking a particular action at a specific state [83–90]. Most of the literature tends to identify model-based approaches as the third cluster of this classification, but we reckon it is a higher-level distinction. Therefore, it has been discussed beforehand.

Several methods have been proposed to solve these problems, and they are typically divided into Monte Carlo (MC) methods, dynamic programming (DP), and temporal difference (TD) methods. Nevertheless, the latter methods are typically blended in the most successful RL algorithms, meaning that the best-performing algorithms combine the advantage of MC, DP,

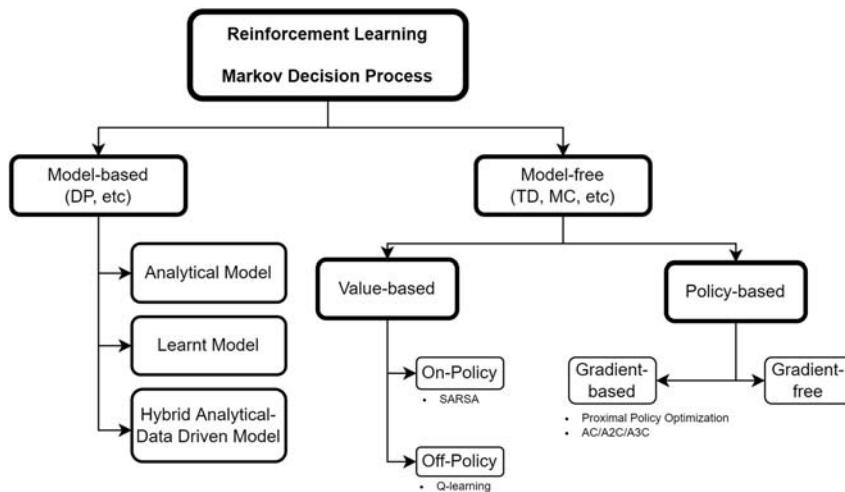


Figure 15.43 Taxonomy of reinforcement learning algorithms.

and TD methods. Although the spectrum of RL algorithms is very wide and fast growing, we try to categorize them hierarchically in Fig. 15.43. Later in the chapter, the most used RL algorithms in space research are presented.

A remarkable take-home message is that reinforcement learning has originally been developed for discrete MDP. This limitation, which is not solved for many RL methods, implies the necessity of discretizing the problem into a system with finite number of actions. This is sometimes hard to grasp in a domain in which the variables are typically continuous in space and time (think about the states or the control action) or often discretized in time for implementation. Thus, the application of reinforcement learning requires smart ways to treat the problem and dedicated recasting of the problem itself. A further clustering for the methods is presented in Table 15.3 and described below:

Table 15.3 Value-based versus policy-based.

Value-based	Policy-based
Q-learning, value iteration	Advantage actor + critic, cross-entropy methods, deep deterministic policy gradient, proximal policy optimization
Explicit exploration	Innate exploration and stochasticity
Evaluates states and actions	Easier for continuous control
Easier to train off-policy	Compatible with supervised learning

- *Value-based methods.* These methods seek to find optimal value function V and action-value function Q , from which the optimal policy π is directly derived.
- *Policy-based methods.* These methods search for the optimal policy π^* directly, which provides a feasible framework for continuous control.

An additional distinction in reinforcement learning is on-policy and off-policy. On-policy methods attempt to evaluate or improve the policy that is used to make decisions during training; whereas off-policy methods evaluate or improve a policy different from the one used to generate the data, i.e., the experience.

Deep reinforcement learning algorithms

Q-Learning and deep Q-learning network

Q-learning and its DL extension is among the most used algorithms for reinforcement learning. It is a TD, value-based method that considers not only the state but also the Q function, or so-called action-value function. As mentioned before, the Q function, or action-value function, represents the expected return (total reward), given a certain state and action. Q-learning approaches are based on the optimization of the action-value function Q , based on the Bellman optimality equation for Q :

$$Q^*(x_k, a_k) = \max_{\pi} Q(x_k, a_k)$$

$$\pi^* = \operatorname{argmax}_{\pi} Q^*(x_k, a_k)$$

$$Q^*(x_k, a_k) = E[r(x_k, a_k) + \gamma \max_{a_{k+1}} Q(x_{k+1}, a_{k+1})]$$

Hence, the method resolves in finding and approximating the optimal action-value function from which to derive the optimal policy. The original method was developed by Watkins [91]: given a certain learning rate η , the next estimation of the action-value function can be calculated as:

$$Q_{k+1} = (1 - \eta) Q_k(x_k, a_k) + \eta [r(x_k, a_k) + \gamma Q_k(x_{k+1}, a_k)]$$

It is important to remark that Q-learning can only deal with MDPs with discretized states and action spaces. Q-learning is a simple yet effective algorithm to create a tabular guidance for our spacecraft. Nevertheless, if the number of states and actions increases, the tabular format to encode the learning becomes rapidly intractable [92]. Deep Q-learning network is the extension of Q-learning in which the Q -function is encapsulated in an ANN. The experience is built up by the agent while playing different

episodes, meaning repeating the same task over and over to become acquainted with the environment. During such episodes (we can think of an episode as a single simulation of a spacecraft task: landing, orbital control, etc.), the agent takes actions according to an ϵ -greedy policy. The ϵ -greedy policy yields a random action with probability ϵ , otherwise it takes the action that maximizes the Q-value function. The ϵ value is typically varied throughout the episodes, in such a way that the agent begins to rely on the Q-function only once the ANN starts approximating it well. Indeed, the most common ϵ evolution is an exponential decay:

$$\epsilon = \epsilon_f + (\epsilon_i - \epsilon_f) e^{-\gamma \Delta t}$$

where ϵ_i , ϵ_f are the initial and final values of the probability, γ is the decay rate, and Δt is the current time step.

Pragmatically, this means that the agent randomly explores the environment at the beginning of its experience, whereas it exploits the knowledge at the end of its experience. It is rather clear that the choice of ϵ is a critical step in setting up the RL framework.

The main components of the deep Q-network (DQN) architecture are (see Fig. 15.44):

- *Q-network*. The network represents the agent action–value function. This network is constantly updated, and it is responsible to yield the optimal action in a given state. Formally, it is the extension of the table for Q-learning. At each time step, the Q-network outputs the Q-value for a given state and action.
- *Target network*. The network is used to approximate the Q-value, but it is updated only once in a while. At the beginning, the target network is equal to the Q-network: the latter is updated at each time step, whereas the former is reset equal to the Q-network only after a given number of

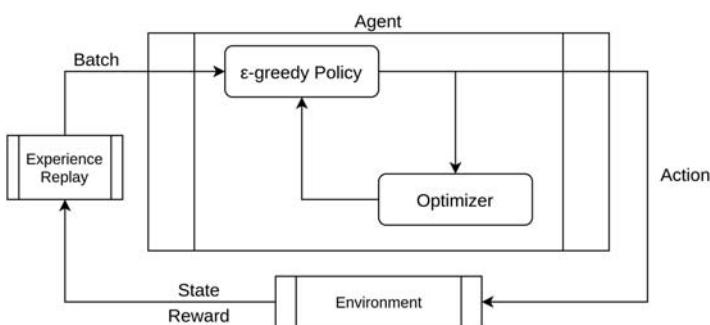


Figure 15.44 DQN schematic architecture.

time steps. At each time step, the target network takes the next state and predicts the best Q-value out of all actions that could be undertaken from that given state.

- *Experience replay.* The experience replay represents the storage of training data. The agent takes an action and receives a reward. Such information tuple is saved as an experience. Why do we need to store experience if we can update the training at each action undertaken? The reason why it is necessary to create a memory database is due to the inherent training procedure of ANNs. These mathematical architectures require a batch of data to train efficiently and effectively; thus, it is beneficial that each time the network updates itself, it does it on a batch of experience rather than a single state-action-reward value.

Advantage actor-critic network

The actor-critic algorithm is a policy-based architecture that aims at optimizing directly the policy to be followed. As stated, these methods are well suited for continuous and stochastic problems. Furthermore, they usually converge faster than value-based methods. The actor-critic method tries to merge the two types of reinforcement learning methodologies, synthesizing the benefits and limiting the drawbacks of both. The breakthrough idea of actor-critic is to split the model in two: one for computing an action based on a state and another one to estimate the state-action value function (Q-function).

The actor takes as input the state and outputs the best action. It essentially controls how the agent behaves by learning the optimal policy (policy-based). The critic, on the other hand, evaluates the action by computing the value function (value-based). Those two models participate in a game where they both get better in their own role as the time passes. Allegedly, the result is that both the actor and the critic increase their performance as the learning proceeds.

In synthesis, the two main components (see Fig. 15.45) act as follows:

- *Actor.* The actor is a function approximator, most commonly an ANN, and its task is to produce the best action for a given state. Indeed, the agent takes as input the state and yields the optimal action (based on the current training status) as output. Overall, it controls the agent behavior by learning the optimal policy.
- *Critic.* The critic is again a function approximator, most commonly an ANN, which takes as input the environment and the action by the actor, concatenates them, and outputs the action value (Q-value) for the given pair. Basically, it evaluates the action taken by the agent while computing the action-value function.

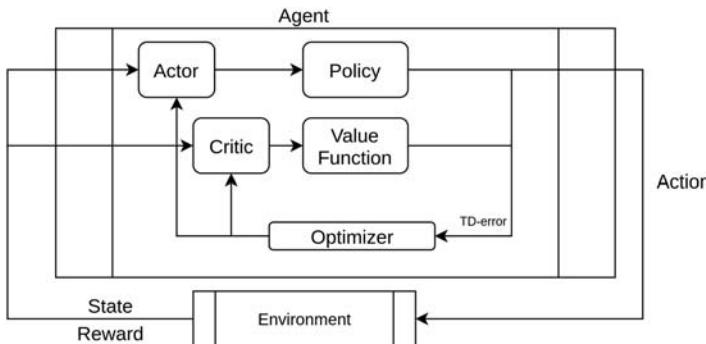


Figure 15.45 A2C schematic architecture.

The training of the two networks is performed separately and exploits the gradient ascent to update the weights and biases of both the actor and the critic network. Then, during the training process, the actor learns to produce better actions while the critic learns to evaluate better those actions. The update can happen both at the end of each time step, in this case, it is defined as TD-0 learning, or at the end of the episode, TD-n learning.

To improve the actor-critic method, two more algorithms have been developed: A2C – Advantage Actor Critic and A3C – Asynchronous Advantage Actor Critic. Regarding the A2C, a new element – the advantage value – is introduced subtracting from the action-value function the value function.

$$A(x_k, a_k) = Q_w(s_k, a_k) - V_w(x_k, a_k)$$

The advantage function represents how much better an action is compared to all the others at a given state; it is different from the value function that instead captures how good it is to be at this state. Exploiting the Bellman equation, it is possible to write the advantage function $A(x, a)$ only in dependence of the value-function:

$$A(x_k, a_k) = E[rt + 1 + \gamma V(x_{k+1})] - V_v(x_k) = r_{k+1} + \gamma V_v(x_{k+1}) - V_v(x_k)$$

In this way, the critic network, instead of learning directly the Q-values, learns the A-values; it means that the evaluation of an action is not only based on how good it is but also based on how much better it can be. The direct consequence is the reduction of the high variance of the policy at each update and, therefore, the stabilization of the entire learning model. As mentioned before, the parameters update may happen in two moments. As TD-0 is defined, the A2C algorithm updates the networks parameters at

each time step; this kind of learning is an on-policy learning because the behavior policy is equal to the update policy. Instead, here as TD-n is defined, the A2C algorithm in which the optimization step is taken only at the end of each episode. In this case, the learning is off-policy because the agent learns when the episode is concluded, and therefore the update and behavior policies are different.

The A3C algorithm was released by DeepMind in 2016 [93]; thanks to its simplicity, robustness, and speed for standard RL tasks, A3C made policy gradients and methods such as DQN obsolete. The difference with A2C is the addition of the asynchronous part: it is characterized by multiple independent agents, where each is defined by its own parameters, interacts with a different copy of the environment, and does it in parallel to the others. In this way, they can explore a bigger part of the state-action space in much less time. The different networks are trained in parallel and update periodically a global network. The term asynchronous comes from the fact that the update does not happen simultaneously, and when it happens, all the agents reset their parameters to those of the global network; then, they continue their independent exploration and training until the next update. As can be seen, the information not only is shared from the agents to the global network but also among the agents themselves, thus leading to a better and more complete results.

Proximal policy optimization

The proximal policy optimization (PPO) algorithm was introduced by the OpenAI team in 2017 and quickly became one of the most popular RL methods usurping the deep Q-learning method. It involves collecting a small batch of experiences interacting with the environment and using that batch to update its decision-making policy. Once the policy is updated with this batch, the experiences are thrown away and a newer batch is collected with the newly updated policy. This is the reason why it is an “on-policy learning” approach where the experience samples collected are only useful for updating the current policy once.

The key contribution of PPO is ensuring that a new update of the policy does not change it too much from the previous policy. This leads to less variance in training at the cost of some bias but ensures smoother training and also makes sure the agent does not go down an unrecoverable path of taking senseless actions.

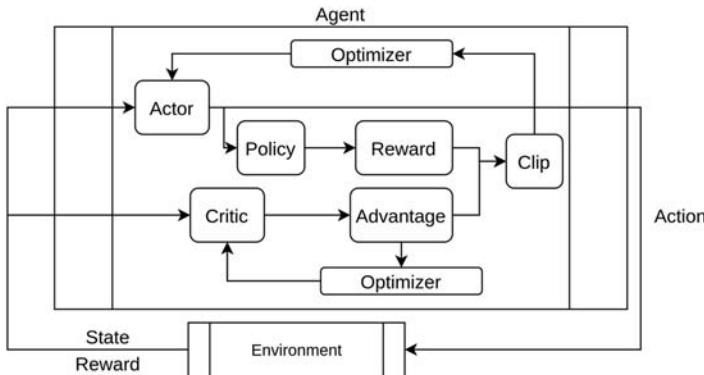


Figure 15.46 PPO schematic architecture.

Although a detailed explanation of the method is outside the scope of this book, a glimpse on the implementation highlights, especially in the space domain is given hereby [85,87,90].

The PPO method is derived from the trust region policy optimization optimization process by accounting for the policy adjustment constraint with a clipped objective function. The objective function used with PPO can be expressed in terms of the probability ratio given by:

$$p_k(\theta) = \frac{\pi_\theta(u_k|x_k)}{\pi_{\theta_{old}}(u_k|x_k)}$$

where θ indicated the hyperparameters of the policy estimate. The actual PPO objective can be written using a clipping function, a wrapper mathematical function that maps a series of data on a bounded domain:

$$J(\theta) = E[min[p_k(\theta), clip(p_k(\theta), 1 - \epsilon, 1 + \epsilon)]A^\pi(x_k, u_k)]$$

This clipped objective function ensures that the policy does not change drastically between updates. In practice, policy gradient algorithms update the policy using a batch of trajectories collected by interaction with the environment. A schematic of the PPO architecture is shown in Fig. 15.46.

Model-based reinforcement Learning

Model-based reinforcement learning techniques may be the easiest concept to grasp for engineers working in the space domain. Indeed, as mentioned at the beginning of the section, such method tries to exploit, build, or learn a model of the environment (what we generally call dynamics) in order to generate predictive control and guidance based on the learnt representation

of the surrounding world [13, 94–99]. In general, we can detect two important features for such approach:

- Predictive models can generalize well enough to generate experiences based on the estimated model rather than the true model.
- Propagations of initial errors inherently present in the learnt environment make long-horizon model prediction unreliable.

The concept of model-based reinforcement learning represents the bridging idea that connects classical approaches such as optimal control to reinforcement learning. Indeed, if we think the *reward* as a *cost function* and the actions as control inputs, we can clearly understand how the two paradigms refer to the same problem.

Inverse reinforcement learning

Inverse reinforcement learning is a task and method of learning an agent's objectives, values, or rewards by observing its behavior. In more traditional terms, this means estimating the cost function or the future state (translational or attitude) trajectory of an observed spacecraft only processing measurements. It is important to remark that, in dynamical system theory (cfr. Appendix A2), the term trajectory represents the set of points in state space that are the future states resulting from a given initial state. This observation strengthens the analogy between reinforcement learning and optimal control, and consequently between inverse reinforcement learning and inverse optimal control.

In general, one can look at inverse reinforcement learning as a framework answering the following [83]:

1. Model the expert's observed behavior as the solution of an MDP whose reward function is not known.
2. Initialize the parameterized form of the reward function using any given features (linearly weighted sum of feature values, distribution over rewards, or other).
3. Solve the MDP with the current reward function to generate the learned behavior or policy.
4. Update the optimization parameters to minimize the divergence between the observed behavior (or policy) and the learned behavior (policy).
5. Repeat the previous two steps till the divergence is reduced to a desired level.

A very brief introduction to two of the most common methods, namely feature-matching approach and maximum entropy is hereby given.

Feature-matching approaches

In general, the cost function of an optimal control problem for trajectory planning in the horizon (t, T) can be described as the summation of a running intermediate cost and the terminal state penalty. The cost function can be recast into a feature-based expression where the cost is a linear combination of f nonlinear features. In principle, each state-control pair can represent a feature. In this method, the optimizer is described as an optimization over a linear combination of features ϕ , which represent cumulative cost along feasible trajectories and terminal state penalty. The feasible trajectories are those respecting optimization constraint and dynamics: the set of state-control pairs represents a policy π , borrowing a term from the reinforcement learning world. Using such approach, the cost function can be rewritten [99]:

$$\begin{aligned} \mathbf{w} &= [w_1, w_2, \dots, w_f, w_T]^T, \boldsymbol{\mu}(\pi) = \left[\begin{array}{c} \sum_t^{T-1} \phi(\mathbf{x}_t, \mathbf{u}_t) \}_{1, \dots, f} \\ \phi_T(\mathbf{x}_T) \end{array} \right] \rightarrow \text{cost: } \mathcal{J} \\ &= \mathbf{w}^T \cdot \boldsymbol{\mu}(\pi) \end{aligned} \quad (15.9)$$

The above formulation is necessary to introduce and discuss the feature-matching approach for inverse reinforcement learning.

The concept of inverse reinforcement learning is to estimate a cost function that delivers an optimal trajectory compatible with an *expert* demonstrated trajectory, called $\tilde{\gamma} = \{(\mathbf{x}_t, \mathbf{u}_t)\}_t^T$ for simplicity. Let us assume the example where the demonstrated trajectory is the reconfiguration path followed by the neighboring agents of a spacecraft formation flying mission. The estimated cost function translates into trajectories through the application of an optimal control problem. Such optimal control problem can be thought as the guidance profile of each agent. It is important to note that the expert cost function, parametric in the set of features, is not known, but we assume the demonstration to be optimal for the estimated one. Each trajectory is generated by a policy π , which characterizes the state-control pairs at each instant in time, ideally. In detail, given a set of N_o observations of the demonstrated trajectory $\mathbf{D}_{\tilde{\gamma}} = \{(\mathbf{x}_i, \mathbf{u}_i)\}_{i=1}^{N_o}$, we need to find a cost function \mathcal{J} , under which the demonstrated trajectory looks optimal according to the estimated cost function. It means that the demonstrated trajectory is the result of an optimization of a certain cost function

that we want to estimate, namely the *expert cost function*. The feature-matching approach solves for the weights in Eq. (15.1) by attempting to match the cumulative feature cost demonstrated by the *expert* under the optimal policy $\tilde{\pi}$ and the policy π^* based on the estimated cost function.

The FMA can be expressed in compact form:

$$\mathcal{J}\left(\tilde{\gamma}|\tilde{\pi}\right) < \mathcal{J}(\gamma|\pi) \rightarrow \mathbf{w}^T \cdot \boldsymbol{\mu}\left(\tilde{\pi}\right) < \mathbf{w}^T \cdot \boldsymbol{\mu}(\pi), \forall \gamma \quad (15.10)$$

where it is stated that the demonstrated trajectory is optimal with respect to the estimated cost function.

The demonstrated trajectory owns significant information about the structure of the cost function; hence, the algorithm significantly benefits if the discrepancy between the estimated optimal trajectory and the expert one is integrated in the optimization [94–96,100].

Loss augmentation represents a cost gap structured margin:

$$\mathcal{L}_{\tilde{\gamma}}(\pi) = \sum_{i=1}^{N_o} \left\| \tilde{\mathbf{x}}_i - \mathbf{x}_i \right\|^2.$$

By inserting the loss augmentation in Eq. (15.10), we obtain the expression for the FMA for IRL:

$$\begin{aligned} & \min_w \|\mathbf{w}\|^2 \\ & \text{subject to } \mathbf{w}^T \cdot \boldsymbol{\mu}\left(\tilde{\pi}\right) < \mathbf{w}^T \cdot \boldsymbol{\mu}(\pi) - \mathcal{L}_{\tilde{\gamma}}(\pi) \end{aligned} \quad (15.11)$$

The form of Eq. (15.11) represents a convex optimization. Nevertheless, the set of policies the algorithm sweeps is theoretically not finite. This makes the optimization intractable, in particular, for computational constraints of several interesting applications in spacecraft GNC. We may suppose that there exists a policy π^* that minimizes the right-hand side of the constraints in Eq. (15.11). Then, the constraints in Eq. (15.11) can be written as $\mathbf{w}^T \boldsymbol{\mu}(\pi) < \min_{\pi} \mathbf{w}^T \boldsymbol{\mu}(\pi) - \mathcal{L}_{\tilde{\gamma}}(\pi)$ without any loss of generality.

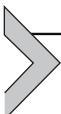
We can place the constraint into the cost function. This yields an unconstrained optimization, which can be solved using gradient-based algorithms:

$$\min_w \frac{\eta}{2} \|\mathbf{w}\|^2 + \left(\mathbf{w}^T \boldsymbol{\mu}\left(\tilde{\pi}\right) - \min_{\pi} \left\{ \mathbf{w}^T \boldsymbol{\mu}(\pi) - \mathcal{L}_{\tilde{\gamma}}(\pi) \right\} \right)$$

where η is a user-defined coefficient. The FMA-IRL is a nested optimization problem. The outer loop is unconstrained, whereas the inner loop, which is a path-planning optimization, may be constrained, both linearly and nonlinearly.

Maximum entropy

IRL is essentially an ill-posed problem because multiple reward functions can explain the expert's behavior. The maximum margin utilized in the feature-matching approach introduces a bias into the learned reward function. To avoid this bias, multiple methods take recourse to the maximum entropy principle [89,91] to obtain a distribution over behaviors, parameterized by the reward function weights. According to this principle, the distribution that maximizes the entropy makes minimal commitments beyond the constraints and is least wrong. In general terms, the methods utilized in literature differ in the distribution whose entropy is used to generate the predictions. The most common selection of methods covers the entropy distribution over trajectories (demonstrated observations) or distribution over policies [83].



AI use cases

This section presents a set of applicative scenarios at research level. The aim of the section is to demonstrate the development of advanced and innovative GNC systems arising from the fundamental content present earlier. The reader is advised to go through this section as if it was a research literature survey on innovative AI-based GNC systems.

Neural dynamics reconstruction through neural networks

The capability of using ANN to approximate the underlying dynamics of a spacecraft is used to enhance the on-board model accuracy and flexibility to provide the spacecraft with a higher degree of autonomy. There are different approaches that could be adopted to tackle the system identification and dynamics reconstruction task. In the following section, the three analyzed methods are described. The three methods employ the ANN model at different integration levels. The different approaches can be clustered in:

- Fully neural dynamics learning.
- Dynamics acceleration reconstruction.
- Parametric dynamics reconstruction.

Fully neural dynamics learning

The dynamical model of a system delivers the derivative of the system state, given the actual system state and external input. Such input–output structure can be fully approximated by an ANN model. The dynamics are entirely encapsulated in the weights and biases of the ANN. The neural network is stimulated by the actual state and the external output. In turn, the time derivative of the state, or simply the system state at the next discretization step, is yielded as output, as shown in Fig. 15.47:

$$\dot{\mathbf{x}} = \mathbf{N}(\mathbf{x}, \mathbf{u}) \rightarrow \mathbf{x}_{k+1} = \tilde{\mathbf{N}}(\mathbf{x}_k, \mathbf{u}_k)$$

The method relies on the universal approximation theorem since it assumes that there exists an ANN that approximates the dynamical function with a predefined approximation error. The training set is simply composed of input–output pairs, where the input is a stacked vector of the system state and control vectors. The dynamics reconstruction based solely on ANNs largely benefits from the employment of RNNs, rather than simpler feed-forward networks.

The fully neural dynamics learning employs an ANN for encapsulating the whole dynamics reconstruction. Let us assume having a system that evolves according to the following equation:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

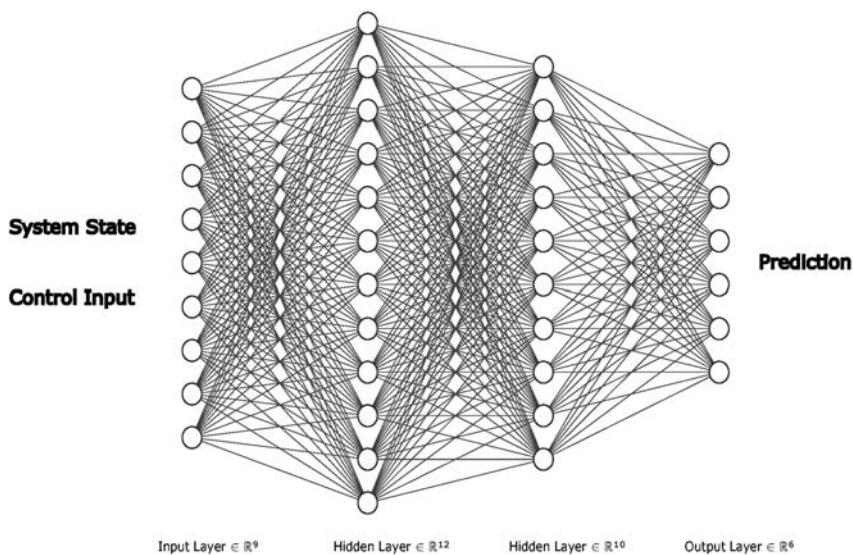


Figure 15.47 Two-layer MLP for dynamics identification.

Furthermore, it is assumed that the observation is equal to the state for the sake of simplicity. The algorithm can easily be extended to different measurement models by introducing the measurement function $h(x)$ or its linear matrix version H . The system dynamics can be learnt using an artificial RNN trained by the standard backpropagation algorithm [13]. One effective strategy is to leverage the physics of the problem in order to obtain a representation, which needs solely some parameters to be fit. Another approach is to couple the neural network with an estimation algorithm to reconstruct only the perturbation terms, taking as basis the linearized natural dynamics. For this example, an RNN is chosen as the neural network architecture. Its simple architecture and brief evaluation time make it a suitable architecture for on-board applications. Recurrent networks have the capability of handling time-series data efficiently. The connections between neurons form a directed graph, which allows an internal state memory. This enables the network to exhibit temporal dynamic behaviors. When dealing with dynamics identification, it is crucial to exploit the temporal evolution of the states; hence, RNN shows superior performances with respect to MLP. In detail, two recurrent networks are proposed to tackle the system identification problem, cfr. Chapter 15 – Modern Spacecraft GNC – AI in space – Introduction. Namely:

- Layer-recurrent neural network.
- Nonlinear autoregressive network with exogenous inputs (NARX).

It is important to remark that the NARX model uses control action as inputs and state as output, given a certain n-delay of the training data [13]. The NARX network is particularly suited for the dynamics reconstruction task, being able to make predictions when used in a closed-loop architecture. Although less performant than RNN, the system dynamics can also be learnt using a feedforward neural network trained by standard backpropagation algorithm [96]. In this example, a two-layer network is employed, namely one hidden layer as sketched in Fig. 15.47. A hyperbolic tangent is used as the activation function for the hidden layer, whereas a rectified linear unit (ReLU) is used as a neural activation function for the output layer.

The dynamics of the spacecraft can be learnt as a discrete model. The downside of such strategy, especially in terms of implementation is that the ANN is associated with a fundamental time step of discretization. In an operative scenario, this could be the sampling frequency of the navigation system. Nevertheless, it could be the case in which the navigation measurements frequency is much higher than the required planning one.

This generates a discrepancy between the model to be learned and the model employed in trajectory generation. However, a too fine sampling in a restricted region negatively affects the generalization capability of the network. The dynamical model can be represented as:

$$\mathbf{x}_{k+1} = f_{T_s}(\mathbf{x}_k, \mathbf{u}_k)$$

where the transition matrix is associated to the sampling time T_s , as stated. These considerations fall into a more fundamental discussion on the importance of dataset generation for DL purposes. As described in Chapter 15 – Modern Spacecraft GNC – AI in space – Introduction, the whole ML and DL domain is data-driven. In simple words, this means that the better the data (samples, distribution, generalization, etc.), the better the performance we could expect.

The model is learned using backpropagation. Let us adopt the well-established Levenberg–Marquardt algorithm for minimizing:

$$\min_w \sum_i \|\tilde{N}_T(\mathbf{x}, \mathbf{u}, w) - \mathbf{y}_{k+1}\|^2$$

where \tilde{N} is the ANN model (function approximator) at the current learning step, hence the dependency on the weights w . The vector \mathbf{y} is the observation vector.

Dynamics acceleration reconstruction

The second method uses the capability of the ANNs to approximate an unknown function. It is wise to exploit every analytical knowledge we may have of the environment. Nevertheless, most of the time, the analytical models encompass linearization and do not model perturbations, either because they are analytically complex or simply unknown. For this reason, in this example, a radial basis function neural network aided adaptive extended Kalman filter (RBFNN-AEKF) for state and disturbance estimation is developed. RBFNNs are selected for their simple structure and suitability for fast online training. The neural network estimates the unmodeled terms which are fed to the EKF as an additional term to the state and covariance prediction step. Finally, a recursive form of the adaptive EKF is employed to limit the overall computational cost. The idea is to combine classical techniques with modern AI-based ones by the combination of mismodeling estimation by the RBFNN together with the adaptive formulation of the EKF, providing a robust, accurate, and computationally

efficient navigation filter that can be initialized and run on-board. In brief, the method can be summarized into three keypoints:

- The AI filter uses the RBFNN to learn and output an estimate of the disturbance/mismodeled terms that is used in the EKF to deliver a better predicted estimate.
- The robustness of the filter is guaranteed through the adaptation step. The combination of an RBFNN and a filter can lead to a very wrong state estimation if the RBFNN estimates diverge or converge to a wrong value; hence, the adaptivity guarantees the robustness of the navigation algorithm.
- The RBFNN learning is fully performed online. This means that no prior knowledge or learning must be performed beforehand. This dramatically increases the flexibility of the algorithm.

The neural network estimates the disturbances acting on the system, which are then adjunct in the prediction step of the filter (cfr. [Chapter 9](#) – Navigation). The filter architecture is sketched [Fig. 15.48](#). The innovation term is used to carry out the adaptivity task. Whereas, the residual term, considering the estimation state at step k , is fed into the online learning algorithm of the network's weights. The system dynamics, considering the process noise, is assumed to be described as:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) + \mathbf{w}$$

Alongside, the measurements are assumed to be perturbed by white noise as:

$$\mathbf{z}_{\text{meas}} = \mathbf{I}_n \mathbf{x} + \mathbf{v}$$

Normally, an observation function, often nonlinear, is introduced, as explained in [Chapter 9](#) – Navigation. In this example, we assume that

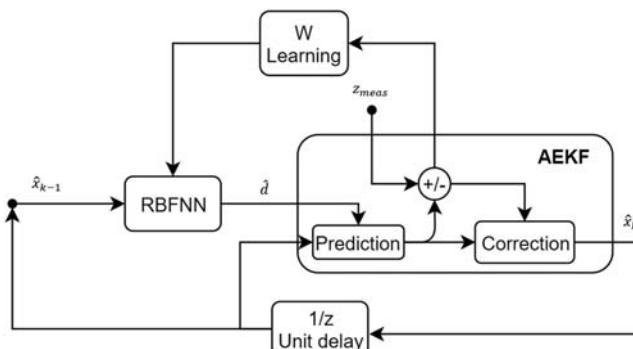


Figure 15.48 Schematics of ANN-aided adaptive extended Kalman filter.

the measurement matrix $\mathbf{H} = \mathbf{I}_n$ or, more general, the measurement model $h(\mathbf{x}) = \mathbf{x}$. In other words, the state is assumed to be completely observable for the sake of derivation, but the approach is applicable to partially observable state and to nonlinear measurement models.

As explained in [Chapter 15 – Modern GNC – AI in space](#), the RBFNN is a popular network topology [14], which has the capability of universal approximation. Due to its simple structure and much quicker learning process, it stands out compared to the classic MLP, especially for function approximation applications. The neurons of RBF are nonlinear Gaussian function; hence, a shallow network can be used with the same results of MLP. Hence, we will employ a light network for this example, consisting of one input, one hidden, and one output layer. The input layer processes the state vector $\hat{\mathbf{x}}_{k-1} = [x_1, x_2, \dots, x_n]^T$, where $k - 1$ is the time instant. The hidden layer performs a nonlinear mapping of the input, whereas the output layer is a linear combination of the nonlinear hidden neurons transformed into the resultant output space. The output space is the disturbance vector $\hat{\mathbf{d}} = [d_1, d_2, \dots, d_n]^T$. An RBFNN is used to estimate the unmodeled disturbances, as well as the nonlinearities present in the system dynamics. The network has a three-layers structure, comprising an input, output, and hidden layer. For the sake of derivation, we call $\mathbf{x} \in R^n$ the input vector.

It is hereby remarked that the vector \mathbf{x} is employed to derive the network structure: in the following sections, the distinction between state vector and estimated state will be described and treated accordingly.

Similar to the input vector, $\Phi \in R^m$ is the hidden layer vector and i the associated index, \mathbf{d} is the output vector and i the associated index. Essentially, the hidden layer evaluates a set of m RBFs which are chosen as centered-Gaussian expression:

$$\Phi(\mathbf{x}) = e^{-\eta(||\mathbf{x} - \mathbf{c}_i||)^2}, \forall i = 1 : m \quad (15.12)$$

where m is the number of neurons and \mathbf{c}_i is the randomly selected center for neuron i . The number of neurons m is a user-defined parameter: its value is application-dependent, and it shall be selected by trading-off the reconstruction accuracy and the computational time. The same consideration holds for the parameter η , which impacts the shape of the Gaussian functions. As already discussed, a high value for η sharpens the Gaussian bell shape, whereas a low value spreads it on the real space. On one hand, a narrow Gaussian function increases the responsiveness of the RBF network,

on the other hand, in case of limited overlapping of the neuronal functions due to too narrow Gaussian bells, the output of the network vanishes. Hence, ideally, the parameter η is selected based on the order of magnitude of the exponential argument in Eq. (15.12). The output of the neural network hidden layer, namely the radial functions evaluation, is normalized.

In a compact form, the output of the network can be expressed as:

$$d(\mathbf{x}) = \mathbf{W}^T \Phi(\mathbf{x})$$

where $\mathbf{W} = [w_{il}]$ for $i = 1, \dots, m$ and $l = 1, \dots, j$ is the trained weight matrix, and $\Phi(\mathbf{x}) = [\Phi_1(\mathbf{x}) \Phi_2(\mathbf{x}) \dots \Phi_m(\mathbf{x})]^T$ is the vector containing the output of the RBFs, evaluated at the current system state. The dynamical model can be described by a set of nonlinear differential equations:

$$\dot{\mathbf{x}} = f(\mathbf{x}) + \mathbf{d}_{ext} \quad (15.13)$$

where the term \mathbf{d}_{ext} is representative of the unknown disturbance acceleration that is added to the known dynamics function $f(\mathbf{x})$. In particular, the disturbance term gathers the contribution of all the environmental perturbations and unmodeled terms. These uncertainties need to be estimated online. Hence, an online learning algorithm, which drives the update of the weights, is required. The weights update law is derived to guarantee the stability of the estimation algorithm and neural dynamics, hereby defined as the evolution of the weight matrix in time.

In the following mathematical derivation, we make use of the universal approximation theorem for neural networks that guarantees the existence of a set of ideal weights \mathbf{W} that approximates a function with a bounded arbitrary approximation error. Such weights are unknown; hence, the algorithm is designed to obtain an estimate $\widehat{\mathbf{W}}$ of the ideal weights by performing on-line learning. The neural network learning algorithm relies on the estimation error dynamics, targeting convergence, and stability of the estimated weights matrix $\widehat{\mathbf{W}}$ evolution toward the ideal weights and the error \mathbf{e} , calculated in the EKF. The symbol $\widehat{\cdot}$ is used to refer to estimated quantities, as described in Chapter 1 — Introduction. To derive the error dynamics, let us assume the actual system dynamics is described by Eq. (15.13), where \mathbf{d}_{ext} is the unknown external disturbance term. The actual system dynamics can be rewritten as the following equation, assuming to include all the nonlinear terms into $d(\mathbf{x})$, which is the vector-valued function equivalent to the RBFNN output vector:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + d(\mathbf{x})$$

where the term $d(\mathbf{x})$ captures all the nonlinearities together with the unknown disturbances external to the system, namely $d(\mathbf{x}) = f(\mathbf{x}) - \mathbf{A}\mathbf{x} + \mathbf{d}_{ext}$. The matrix \mathbf{A} is a stable, potentially time-varying, matrix representing the linear term, if any, of the original dynamics expression in Eq. (15.13).

The expression of the continuous single-step Kalman filter can be written as (cfr. Chapter 9 – Navigation):

$$\frac{d\hat{\mathbf{x}}}{dt} = \mathbf{A} \cdot \hat{\mathbf{x}} + \hat{d}(\hat{\mathbf{x}}) + \mathbf{K}_k \mathbf{H}(\mathbf{x} - \hat{\mathbf{x}})$$

where \hat{d} is the estimated function using the radial-basis function neural network, \mathbf{K}_k is the time-varying gain matrix of the Kalman filter (subscript k stands for the referred time step), and \mathbf{H} is the observation matrix. Consider that the continuous form is employed for the sake of derivation, in fact, the learning rule is then discretized for the actual implementation. The error dynamics can be derived as:

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$$

$$\dot{\mathbf{e}} = \dot{\mathbf{x}} - \frac{d\hat{\mathbf{x}}}{dt} = d(\mathbf{x}) - \hat{d}(\hat{\mathbf{x}}) + (\mathbf{A} - \mathbf{K}_k \mathbf{H})\mathbf{e}$$

Invoking the universal approximation theorem for neural networks, we can assume there exists an ideal approximation of the disturbance term $d(\mathbf{x})$:

$$d(\mathbf{x}) = \mathbf{W}^T \Phi(\mathbf{x}) + \varepsilon$$

where \mathbf{W} is the neural weights matrix, $\Phi(\mathbf{x})$ is the vector-valued function resulting from the evaluation of the Gaussian functions contained in each neuron of the RBFNN, ε is a bounded arbitrary approximation error. Consequently, the error in estimation can be written as:

$$d(\mathbf{x}) - \hat{d}(\hat{\mathbf{x}}) = \mathbf{W}^T \Phi(\mathbf{x}) + \varepsilon - \widehat{\mathbf{W}}^T \Phi(\hat{\mathbf{x}})$$

by adding and subtracting the term $\mathbf{W} = \Phi(\hat{\mathbf{x}})$ and performing few mathematical manipulations, the above equation can be expressed as:

$$\tilde{d} = \widetilde{\mathbf{W}}^T \Phi(\hat{\mathbf{x}}) + \varepsilon_j$$

where $\tilde{d} = d - \hat{d}$, $\widetilde{\mathbf{W}} = \mathbf{W} - \widehat{\mathbf{W}}$, and the bounded term $\varepsilon_j = \varepsilon + \mathbf{W} \cdot [\Phi(\mathbf{x}) - \Phi(\hat{\mathbf{x}})]$. The aim of the learning rule is to drive the dynamics error to zero, as well as forcing the weights to converge to the ideal ones. The weights update rule $\dot{\widetilde{\mathbf{W}}}$ is derived to guarantee the stability and convergence of the estimation algorithm:

$$V = \frac{1}{2} \text{tr} \left(\xi \widetilde{\mathbf{W}}^T \widetilde{\mathbf{W}} \right) + \frac{\eta}{2} \mathbf{e}^T \mathbf{e} \quad (15.14)$$

where $\text{tr}(\cdot)$ is the trace operator, $\xi, \eta > 0$ are user-defined coefficients. The derivative of the Lyapunov function can be written as:

$$\begin{aligned} \dot{V} &= \text{tr} \left(\xi \widetilde{\mathbf{W}}^T \dot{\widetilde{\mathbf{W}}} \right) + \eta \mathbf{e}^T \dot{\mathbf{e}} \\ &= \text{tr} \left(\xi \widetilde{\mathbf{W}}^T \dot{\widetilde{\mathbf{W}}} \right) + \eta \mathbf{e}^T \left(\widetilde{\mathbf{W}}^T \Phi(\hat{\mathbf{x}}) + \varepsilon' + (\mathbf{A} - \mathbf{K}_k \mathbf{H}) \mathbf{e} \right) \\ &= \text{tr} \left(\xi \widetilde{\mathbf{W}}^T \dot{\widetilde{\mathbf{W}}} \right) + \eta \mathbf{e}^T \widetilde{\mathbf{W}}^T \Phi(\hat{\mathbf{x}}) + \eta \mathbf{e}^T \varepsilon' + \eta \mathbf{e}^T (\mathbf{A} - \mathbf{K}_k \mathbf{H}) \mathbf{e} \\ &= \text{tr} \left(\xi \widetilde{\mathbf{W}}^T \dot{\widetilde{\mathbf{W}}} + \eta \widetilde{\mathbf{W}}^T \Phi(\hat{\mathbf{x}}) \mathbf{e}^T \right) + \eta \mathbf{e}^T \varepsilon' + \eta \mathbf{e}^T (\mathbf{A} - \mathbf{K}_k \mathbf{H}) \mathbf{e} \\ &= \text{tr} \left(\widetilde{\mathbf{W}}^T \left(\xi \dot{\widetilde{\mathbf{W}}} + \eta \Phi(\hat{\mathbf{x}}) \mathbf{e}^T \right) \right) + \eta \mathbf{e}^T \varepsilon' + \eta \mathbf{e}^T (\mathbf{A} - \mathbf{K}_k \mathbf{H}) \mathbf{e} < 0 \quad (15.15) \end{aligned}$$

If we recall the manipulation of $\widetilde{\mathbf{W}} = \mathbf{W} - \widehat{\mathbf{W}} \rightarrow \dot{\widetilde{\mathbf{W}}} = -\dot{\widehat{\mathbf{W}}}$, we can derive the update rule for the weights such that the stability and convergence of the estimation is guaranteed by the Lyapunov theorem. The expression reads:

$$\dot{\widehat{\mathbf{W}}} = \frac{\eta^T}{\xi} \Phi(\hat{\mathbf{x}}) \mathbf{e}^T$$

Inserting this derivative $\dot{\widehat{\mathbf{W}}}$, the expression for the derivative of the Lyapunov function in Eq. (15.14) reduces to the stability of the error estimation of the extended Kalman filter. The error term represents the residual

between estimated and actual output of the observed system: in practical terms, the expression is the innovation of the estimation filter. The ε_j term is a bounded term that derives from the universal approximation theorem of ANNs that states that the term ε can be arbitrarily small. In practice, it represents an upper boundary for the derivative of the Lyapunov function.

In case of linear systems, the term $(\mathbf{A}\mathbf{K}_k\mathbf{H})$ grants asymptotic stability of the Kalman filter if \mathbf{A} is reachable and \mathbf{H} is observable. In case of nonlinear systems, this is not always true. However, it has been proved that the estimation error of an EKF is exponentially bounded if:

- \mathbf{A} is nonsingular for every $t \geq 0$.
- there exist real constants $p_1, p_2 > 0$ such that $p_1 \cdot \mathbf{I} \leq \mathbf{P}_k \leq p_2 \cdot \mathbf{I}$ where \mathbf{P}_k is the estimated state covariance matrix.
- the initial estimation error satisfies $\|\hat{\mathbf{x}}_0 - \mathbf{x}_0\| < \varepsilon$ and the process and measurements covariance matrices are bounded.

where $\hat{\mathbf{x}}_0$ and \mathbf{x}_0 are the estimated and true state vector at the initial step. Given the EKF asymptotic stability with exponential decaying error under the aforementioned conditions, i.e., the derivative of the Lyapunov function of the estimation error is negative, Eq. (15.15) is verified and hence the stability of the estimator is guaranteed.

The weights update rule can be discretized using a first-order Euler method, assuming the measurements interval is small enough:

$$\widehat{\mathbf{W}}_{k+1} = \psi \widehat{\mathbf{W}}_k + h \dot{\widehat{\mathbf{W}}}_k = \psi \widehat{\mathbf{W}}_k + h \frac{\eta}{\xi} \boldsymbol{\Phi}(\hat{\mathbf{x}}_k) \mathbf{e}_k^T$$

where k is the time step index, ψ is a user-defined relaxation factor, and $h = t_{k+1} - t_k$ is the time interval between two consecutive measurements.

Parametric dynamics reconstruction

The third method is developed under the framework of parameter reconstruction. Basically, the ANN is employed to refine the uncertain parameters of a given dynamical model. This method is particularly suitable when the uncertain environment influences primal system constants (e.g., inertia parameters, spherical harmonics, and drag coefficients). In the example reported in this book, the method is developed using an RNN to estimate the spherical harmonic expansion coefficients of irregular bodies that popu-

late the Solar System. Such method is very fast and computationally light with respect to traditional algorithms for parameters estimation. Moreover, given the physical knowledge of the parameters to be reconstructed, the method has a very promising scientific outcome. For instance, the gravity expansion of asteroids and planets can be approximated online while flying, delivering a rough shape reconstruction of the body. Nevertheless, many applications overlap with the disturbance reconstruction approach in which the disturbance function to approximate is a constant parameter, as assumed in many estimation algorithms, such as the well-established Kalman filter.

For this reason, many researchers used MLP neural networks to carry out the task. For instance, Chu et al. [101] proposed a deep network MLP to estimate inertia parameters. The angular rates and control torque of combined spacecraft are set as the input of a DNN model, and conversely, the inertia tensor is then set as the output. Training the MLP model refers to the process extracting higher abstract feature, i.e., the inertia tensor [101,102].

Another approach to solve the parametric reconstruction is to use an RNN. In particular, HNNs are investigated in Refs. [39,40]. The core of the algorithm is the following: if the dynamical model (cfr. Chapter 10 – Control and Appendix A2) is reformulated into linear-in-parameters form, the identification problem can be reformulated as an optimization problem:

$$\mathbf{y} = \mathbf{A}(\mathbf{x}) \cdot \mathbf{c}$$

In particular, defining a general prediction error $\mathbf{e} = \mathbf{y} - \mathbf{A} \cdot \mathbf{c}^*$, the resulting combinatorial optimization problem is:

$$\min_C \left\{ \sup_t \left(\frac{1}{2} \mathbf{e}^T \cdot \mathbf{e} \right) \right\}$$

where \mathbf{y} typically corresponds to measurements, \mathbf{A} is the linear-in-parameter matrix, and \mathbf{c}^* is the estimated parameter vector [40].

The HNN, as presented in Chapter 15 – Modern Spacecraft GNC – AI in space—Introduction can be used to solve these types of combinatorial problems. Recalling the network fundamental equation:

$$\frac{d\mathbf{s}}{dt} = \frac{1}{\beta} \mathbf{D}(\mathbf{W}\mathbf{s} + \mathbf{b})$$

where we can exploit the fact that the neural network can be adapted to represent the problem of our interest. Indeed, if we define the weight matrix as $\mathbf{W} = -\mathbf{A}^T \mathbf{A}$ and the bias vector as $\mathbf{b} = \mathbf{W}\mathbf{s}_0 + \mathbf{A}^T \mathbf{y}$, we can establish a direct link between the system identification problem and the dynamics of the network. The application of the HNN to the solution of the optimization problem benefits from the existence of a network Lyapunov function that guarantees stability of the network, cfr. Chapter 15 – Modern GNC – AI in space.

Convolutional neural networks for planetary landing

The use of AI in scenarios for the moon or planetary landing is still at an early stage, and few works exist in literature concerning image-based navigation with AI. The presented solutions are all based on a supervised learning approach. Images with different lighting and surface viewing conditions are used for training. One idea is to substitute classical IP algorithms, providing a first estimate of the lander state (e.g., pose or altitude and position), which can be later refined by means of a navigation filter. Similar approaches have been implemented in the robotic field, where pretrained on a large dataset, they estimate the absolute camera pose in a scene. In a landing scenario, the knowledge of the landing area can be exploited, if available. Therefore, a CNN (cfr. Chapter 15 – Modern Spacecraft GNC – AI in space – Introduction) can be trained with an appropriate dataset of synthetic images of the landing area at different relative poses and illumination conditions. The CNN is used to extract features that are then passed to a fully connected layer, which performs a regression and outputs directly the absolute camera pose. The regression task can be executed by an LSTM. The CNN-LSTM has proven excellent performance and is very well developed for IP and model prediction. The use of a recurrent network brings the advantage of also retrieving time-series information. This can allow estimating also the velocity of the lander. According to an extensive review of the applications, one can make a general distinction between two macromethods:

- *Hybrid approaches.* They utilize CNNs for processing images, extracting features, classifying or regressing the state at the initial condition, but they are always coupled with traditional IP or navigation algorithms (e.g., PnP, feature tracking).
- *End-to-end approaches.* They are developed to complete the whole visual odometry (VO) pipeline, from the image input to the state-estimate output.

As an interesting example, although not directly applied to space systems, the technique presented in Ref. [103] relies on an end-to-end learning for estimating the pose of an unmanned aerial vehicle during landing. In particular, the global position and orientation of the robot is the final output of the AI architecture. The AI system processes two kinds of inputs: images and measurements from an inertial measurement unit (IMU). The architecture comprises a CNN that takes as input streams of images and acts as a feature extractor. Such CNN is built starting from ResNet18, pretrained on the ImageNet dataset. An LSTM processes the IMU measurements, which are available at a higher frequency than images. An intermediate fully connected layer fuses the inertial and visual features coming from the CNN and the LSTM. Then, such vector is passed to the core LSTM, along with the previous hidden state, allowing to model the dynamics and connections between sequences of features. Finally, a fully connected layer maps the feature to the desired pose output.

Similarly, the architecture proposed by Furfarò et al. [104] comprises a CNN and an LSTM. The final output of the AI system is a thrust profile to control the spacecraft landing. The CNN input consists in three subsequent static images. Such choice is motivated by the need of retrieving some dynamical information. The whole VO pipeline has been learnt completely in the work by Wang [105]. The approach proposed exploits a DL system based on a monocular VO algorithm to estimate poses from raw RGB images. Since it is trained and deployed in an end-to-end manner, it infers poses directly from a sequence of raw RGB images without adopting any module of the conventional VO pipeline. The AI system comprises the CNN that automatically learns an effective feature representation for the VO problem, but also a recurrent network, which implicitly models sequential dynamics and relations. The final output is the absolute pose of the vehicle. This architecture differs from the one presented in Ref. [103] because here two consecutive frames are stacked together and only images are considered as inputs.

A hybrid approach, specifically developed for lunar landing is presented by Refs. [106,107] and readapted by Refs. [108,109]. The approach is based on the work by Silburt et al. [110]: a DL approach is used to identify lunar craters, in particular an Unet-CNN, is used for input images segmentation, as shown in Fig. 15.49. Some traditional navigation strategies are based on lunar craters matching; therefore, the AI method is investigated as part of a hybrid approach, as in Refs. [106,107] where a RANSAC-based nearest neighbor algorithm is used for matching the detected craters to database

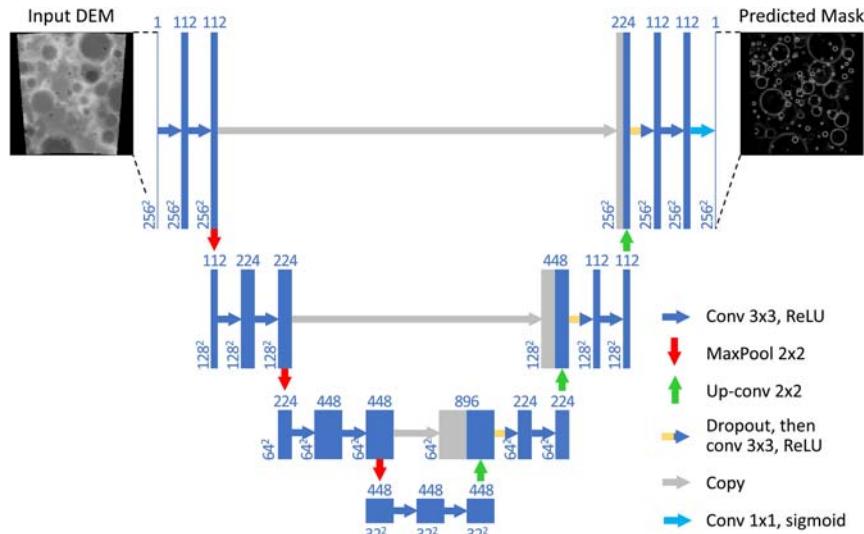


Figure 15.49 Convolutional neural network architecture, based on Unet [110]. Boxes represent cross sections of sets of square feature maps.

ones. The advantage of such a hybrid approach is to combine a crater detection more robust to illumination conditions and the reliability of a traditional pose estimation pipeline. An example of input image and output mask is shown in Fig. 15.50. This is a powerful technique for absolute navigation where database objects can be used for learning. The state estimation requires a navigation filter or feature postprocessing, such as computation of essential matrix, retrieval of relative vectors, to complete the navigation pipeline.

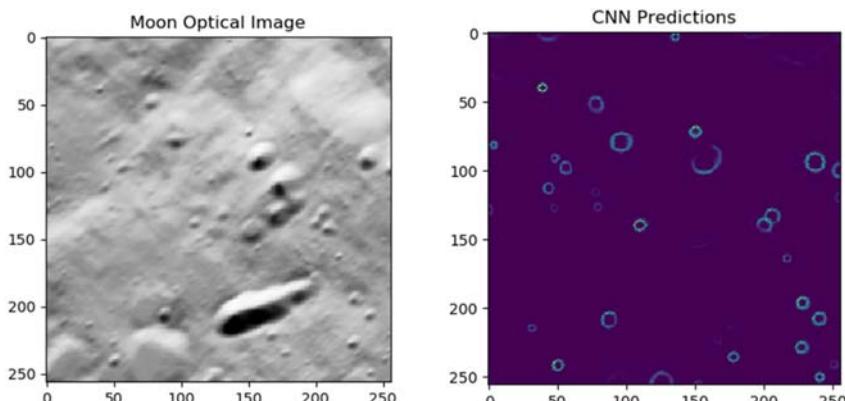


Figure 15.50 Input image and output mask from the CNN.

Table 15.4 Summary of neural-aided methods for optical navigation for planetary and lunar landing. IP stands for image processing and represents the necessity for a hybrid method to be implemented (e.g., matching, PnP).

Type	Accuracy	Training needs	Robustness	Adaptability
CNN + IP – Hybrid	High	Medium	High	Medium
CNN + RNN – E2E	Medium	High	High	High

A qualitative summary of the most promising methods is reported in [Table 15.4](#): Summary of neural-aided methods for optical navigation for planetary and lunar landing. IP stands for image processing and represents the necessity for a hybrid method to be implemented (e.g., matching, PnP).

Deep reinforcement learning for uncooperative objects fly around and planetary exploration

An active research field is to use reinforcement learning and meta-reinforcement learning (meta-RL) to create an adaptive guidance and control system. Deep reinforcement learning has been used to generate autonomous guidance and control during proximity operations and landing trajectories [92,111,112]. Reinforcement learning has been used to generate autonomous trajectory planning for different scopes. Pesce et al. [113], Piccinin et al. [114], and Chan et al. [115] analyzed the autonomous mapping of asteroids using DQN, Neural Fitted Q as value-based methods. Federici et al. [116] proposed an actor-critic PPO framework for real-time optimal spacecraft guidance during terminal rendezvous maneuvers, in presence of both operational constraints and stochastic effects, such as an inaccurate knowledge of the initial spacecraft state and the presence of random in-flight disturbances.

Brandonisio et al. [111] proposed a guidance and control law to perform the inspection of an uncooperative spacecraft. Two of the most common reinforcement learning methods described earlier are used, namely DQN and A2C methods. State-action value functions are approximated using ANNs: in particular, simple MLPs are employed. A discrete action space is maintained, whereas the state space is continuous. Transfer learning (TL) is also applied to facilitate training on more complex tests. One of the TL techniques consists of pretraining the RL agent on a simpler task before training on the main task. In the paper, the various tasks are represented by increasing complexities of the reward models. Many researchers claim a superior performance of policy-based methods. Among those,

PPO and derivatives is one of the most adopted schemes [111,116–118]. To improve the stability and robustness of the agent in different scenario conditions, a formulation of PPO, but not only, exploiting RNNs is commonly exploited. As anticipated in Chapter 15 – Modern GNC – AI in space – Introduction, the capability of recurrent layers to store past states information may strongly affect the agent safe trajectories planning and faster getting to the mission goals. In addition, training an RNN is beneficial to refine the agent’s environmental conditions sensitivity, which works in favor of the agent’s robustness, regardless of the specific operational environment. The work in Ref. [119] proposes a guidance strategy for spacecraft proximity tracking operations leveraging deep reinforcement learning. In Ref. [119], the distributed distributional deep deterministic policy gradient (D4PG) algorithm is used. Such algorithm operates in continuous state and action spaces, and it has a deterministic output. The D4PG algorithm is an extension of the actor-critic algorithm.

Meta-reinforcement learning

The robustness to uncertain spacecraft model and environment is a crucial topic in the development of autonomous systems. This aspect is very challenging when it comes to reinforcement learning methods. Neural networks learn well within the training distributions, but they generally fail when performing extrapolation outside the training distribution [120]. This may pose a risk of instability of the guidance law when the spacecraft experiences states that are outside the training distribution envelope. Several researchers [117,118,120] claim that sampling inefficiency is another weakness of traditional reinforcement learning: a large amount of experience is needed to learn even the simple tasks. In Ref. [121], the authors propose a reinforcement learning framework to control a spacecraft around a small celestial body with unknown gravity field. In particular, the hovering task is investigated. The authors perform a direct policy search with a genetic algorithm to obtain controller policies with high utility. The policy architecture used is a simple MLP.

Recent advancements on meta-RL have aimed at addressing these weaknesses of the traditional framework. Reinforcement meta-learning trains the policy agent on a distribution of environments or MDPs. This forces the agent to experience and learn multiple, and different, situations. Consequently, the system tends to converge faster to quasioptimal solutions.

Recent works claim superior performance of the meta-RL with respect to classical RL when uncertain environments and actuator failures are

considered [117,118,120]. Gaudet et al. [117,118] developed a guidance law based on meta-RL which is able to perform a six degrees-of-freedom mars landing. Moreover, in Ref. [118], meta-RL is used to create a guidance law for hovering on irregularly shaped asteroids using light detection and ranging (LIDAR) sensor data.

In certain works, the different environments are called tasks: the tasks can be thought of the ensemble of potential situations, nominal and nonnominal, one can expect the agent to experience. For instance, in the application of meta-RL for planetary and asteroid landing, the tasks range from landing with engine failures to large mass variation or highly corrupted navigation and unknown dynamics.

In Li [122], a meta-RL framework is employed for relative trajectory planning between spacecraft. The training trajectories are divided as sub-training samples and fake testing samples. The meta-reinforced agent is trained by alternating training and testing phases. To this end, the gradient information of the meta learner is obtained through a combination of the results on the training subsets and the performance on the fake testing samples. The authors in Ref. [122] claim that this approach forces the agent to explicitly take account of the potential testing performance into consideration. In this way, the overfitting phenomenon, potentially arising using few training trajectories, is reduced.

In most of the works, as for traditional reinforcement learning applications, meta-RL policy is optimized using PPO, both the policy and value function implementing recurrent layers in their networks. Using RNNs results in creating agents that can adapt more easily to uncertain environments, yielding a much more robust guidance policy compared with classical reinforcement learning. If we examine a particular scenario such as planetary landing, recalling what was described in Chapter 15 – Modern GNC – AI in space – Introduction, it is easier to understand how recurrent layers result in an adaptive agent. During the training to generate an autonomous landing agent, the next observation depends not only on the state and action but also on the GT agent mass and any external forces acting on the agent at each step. Consequently, during training, the recurrent layers force their hidden states to evolve differently depending on the observations acquired from the environment during the trajectory, which is governed by the actions output by the policy. Specifically, the trained policy’s hidden state captures unobserved information such as external forces or agent mass that are useful in minimizing the cost function. Obviously, this holds for any scenario one could be interested in, even not related to space operations.



AI on-board processors

Modern GNC algorithms, implementing advanced techniques and exploiting computationally expensive AI-based methods, demand the development of dedicated avionics solutions. In fact, a standard processor, being single core, dual-core, or quad-core, will struggle on executions of neuronal networks with many layers, with large connectivity among layers and with increasing data vectors being passed from layer to layer. For this reason, specific AI processing avionics try to boost the utilization of many processing units in parallel with extensive data utilization and hence with large access to memory units in the minimum time possible. In general, it exists an analogy between ML techniques and computer-vision algorithm, using many convolution filters. For this reason, the selection of the architecture and avionics components for AI can follow a similar approach used for vision-based navigation solutions.

First, it's necessary to distinguish between on-ground and on-board platforms. On-ground solutions are used for the training process of the neural network, and, on the other hand, on-board platforms are used for the AI solution execution. In this way, the AI method guarantees high-performance and limited power consumption on-board being extensively trained on-ground, taking advantage of very powerful methodologies that are so energy-hungry in terms of power consumption that could never be selected for on-board executions (not even in rovers with nuclear energy source).

Most of the AI solutions on-ground try to exploit the utilization of GPUs (see [Chapter 13 – On-board implementation](#)). In space, the options for GPUs, with enough environmentally induced failures protection and reduced power consumptions are very hard to find, but it is interesting to consider solutions as soft-core GPUs implemented in FPGA for space. The use of standard GPUs is generally discarded for its use on-board due to their high-power dissipation, but recent developments (e.g., Nvidia Tegra X1 GPU SoC) are trying to provide solutions focused on power efficiency for embedded applications. None of these GPU-based alternatives is yet accepted for in-flight space applications at the time of writing of this book.

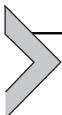
A very interesting deviation from GPUs, with specific objective for using it on Vision or AI solutions, is represented by chips based on multiple VPUs interconnected and combined with LEON processor (e.g., Movidius Myriad and Myriad2). A VPU is a type of microprocessor designed to process visual data and may include interfaces for direct streaming of image

sensor data. A VPU is aimed at accelerating IP and CV algorithms. Due to the efficient vector (parallel) processing architectures and memory layouts that are required to effectively process image data, VPUs are typically also suitable for efficiently implementing ML and AI inference tasks. In fact, the previously mentioned Myriad 2 was designed primarily as a vision processing device, with HW support for various IP functions (e.g., Harris, median and edge filters, up/down scalers), but it also proved to be very effective at running CNNs (i.e., it was chosen by ESA for deployment as the AI inference engine on the HyperScout-2 hyperspectral payload, which forms part of Φ Sat-1 currently in orbit). VPUs are designed to provide ultralow power capabilities, without compromising performance. In general, VPUs are not intended to be the primary processor in a system, but to act as an accelerator, or coprocessor. VPUs are built for parallel processing, and their performance depends on the fact that visual data come in 2D arrays (at least) and that vision functions in general exhibit massive data parallelism.

Other alternative solutions are based on FPGA (see [Chapter 13 – On-board implementation](#)) with specific deployment of convolutional engines in the frame of AI-based IP solutions and interconnections in FPGA logic. These kernels are suited for FPGA implementation, making use of the hard blocks of its embedded DSPs, explicitly built for fast mathematics operations. FPGA offers different advantages over the vastly used GPUs for DL. Both HW solutions allow the possibility to use many cores or instances of function to exploit parallelism. GPUs where first conceived for render and display graphics or video, and for DL deployment might present some limitations. The FPGA is a blank device, and the HW can be programmed by means of the configuration of the programmable logic gates, functions, and routing, and, therefore, it offers a lot of flexibility. It provides excellent performances at the advantage of much lower power consumption than GPUs. Several other functions can also be added, as cryptosecure blocks, or different communication interfaces, or sensor fusion. The FPGA can implement protocols and low-level interfaces layers for multiple standards, and, therefore, it can embed multiple sensors in one-chip interface. Moreover, it provides the extra capability of implementing in the FPGA logic the pre-processing or acquisition synchronization and sensor fusion, which facilitates certain AI applications using multiple inputs from various sensors (e.g., cameras, LIDAR, RADAR, Sun Sensor, or IMU) for the same application. Similarly, the FPGA can manage different memory units, can devote internal FPGA blocks for buffering data, and can implement functions to overcome the I/O bottlenecks problems that might affect AI systems performances where large amount of data is used.

There are different attempts to create FPGA modules reusable for different CNN architectures designs or implementation deployment. Frameworks are being developed to allow an AI engineer to use Python, C, or C++ and typical known AI frameworks at high level to pass to an automatic process inferring the solution onto the deployment HW board, for instance, using high-level synthesis tools to create FPGA solutions. Tools such as Mathworks provide AI framework with a final autocoding steps targeting GPUs or FPGAs. Xilinx, the major FPGA vendor in the world, provide Delphi Deep Neural Network Development Kit (DNNDK) to help AI developers on the deployment of their solutions in Xilinx FPGAs using these DNNDK library.

Few examples are described hereafter. For instance, Xilinx is betting on heterogeneous HW platforms with dedicated AI engines such as the brand-new Versal AI cores Series FPGA [123] and the Vitis SW tool [124] to program them. Nevertheless, these solutions are not offered for space. FINN framework [125] is also provided by Xilinx, more generic than Vitis although only targeting quantized neuronal networks and more as a space exploration of acceleration possibilities in FPGA for the DNN inference. Moreover, as presented in Ref. [126], DNNDK is a full-stack DL SDK for the deep learning processor unit provided now by Xilinx. It provides a full-stack solution for DL inference application development in a unified solution for DNN inference applications by providing pruning, quantization, compilation, optimization, and runtime support. The framework provides a complete set of optimized tool chains, including compression, compilation, and runtime and overcomes one of the high-level programmers' drawbacks by providing a lightweight C/C++ programming application programming interface (API) to make it simple for users without FPGA knowledge to develop DL inference applications by providing a set of lightweight C/C++ APIs while abstracting away the nature of underlying FPGA device.



Innovative techniques for highly autonomous FDIR in GNC applications

This section presents technical solutions and development processes for implementing highly autonomous on-board FDIR systems, with a particular focus on GNC applications. As shown in the [Chapter 11](#) – FDIR Development Approaches in Space Systems, generally speaking, FDIR systems act as supervisory controllers and ensure that the mission objectives and dependability/safety requirements are met, so that spacecraft

(S/C) are protected from failures leading to a service deterioration or even worst to the mission loss [127,128]. The survival of the spacecraft has generally priority over its service availability. FDIR systems monitor and process streams of observations in order to detect, isolate, and, if necessary, react to events and anomalies occurring inside the spacecraft [129].

FDIR system conception, design, implementation, verification, and validation are very complex tasks [127]. They strongly depend on both system-level and operational requirements (such as the S/C operational modes and mission phases [128]). Moreover, there is also a strong connection between FDIR systems and S/C on-board autonomy, which is becoming the key-point in the design of future space missions [130,131].

Today's rapid progress in on-board resources (i.e., memories and computational power) is actually fostering the transfer of FDIR functions from the ground to the flight segment, and thus the enhancement of on-board autonomous failure management capabilities [131,132]. This section describes innovative approaches and solutions for on-board FDIR systems that are likely to be implemented within a short/medium term (or at least industrial efforts and interests driven by the space agencies would prove this statement [137,138]) in order to address the limitations of currently used FDIR solutions [127,128] and the needs of future space missions. First, we discuss the evolution of FDIR systems in the upcoming years and highlight all the limitations of currently used FDIR systems and the needs of future space missions. In particular, model-based and data-driven (also called soft-computing) FDIR systems are introduced. As shown later, they can provide the capabilities of processing anomalous observations in spite of uncertainties and partial observability in order to estimate the system health status and determine the most appropriate corrective action [135,139–141]. After that, the central part of this section presents advanced FDIR solutions for GNC applications. More specifically, we address both model-based and data-driven approaches. The former includes control theory solutions (e.g., observer-based methods) to generate residuals for the failure detection and isolation [139,140]. On the other hand, when no explicit dynamical model is available, system knowledge boils down to historical and/or real-time data measurements, which can be used to, e.g., train classifiers with the aim of assigning newly measured variables to classes representative of healthy or faulty behaviors [139,141]. AI and ML solutions fall in the data-driven category. Finally, we explain the impact of a model-based approach on the FDIR system development, and then we outline the major challenges for the actual industrial implementation of model-based and data-driven FDIR systems.

FDIR system evolution in the next years

FDIR system conception and implementation in a space system can be a rather complex field, since the analysis of S/C failures can extend to very sophisticated considerations. The cause of malfunctions could be one or more failed components, incorrect control actions, or external disturbances, which impact system operations [128,142,143]. In complex, heterogeneous systems with large numbers of interacting physical components, the effects of faults can propagate through the boundaries of many subsystems and can result in diverse fault symptoms. FDIR information processing and physical components are tightly integrated and intertwined; therefore, the structure and properties of physical S/C components determine the functions to be implemented by an FDIR system.

The level of complexity of FDIR systems has been addressed by the space agencies, the industry, and the academia since long time. For instance, in Ref. [144], we can find a list of findings and recommendations emerged during a NASA workshop held in 2008. The following topics were mainly discussed: (i) FDIR system architectures; (ii) FDIR system development practices, processes, and tools; (iii) FDIR system verification and validation. During such workshop, there was a general agreement that FDIR systems in space missions are limited not only by technology but also by a lack of emphasis in both system engineering and programmatic dimensions.

Current FDIR system development processes and technical solutions have shown some relevant technological and development limitations. As for the former, we have to highlight that the lack of proper system-level development approaches causes serious discontinuities throughout all the project phases and hampers the process of a stable and consistent FDIR design [127,144]. In this regard, model-based system engineering (MBSE) paradigm can be of great help for managing complexity, maintaining consistency, and assuring traceability during the overall FDIR system development. MBSE is different from engineering with models. Indeed, MBSE is a system engineering approach centered on evolving system models (and not on documents), which serve as the sole source of truth about the system. It comprises system specification, design, validation, and configuration management [145]. A central role is played by the FDIR system toolset environment, which should provide a seamless support throughout the different FDIR development phases. Tool integration can be facilitated through work on common terminology/taxonomy, metrics, and interface specifications. Ideally, such toolset would be available for early behavioral modeling

of FDIR systems and would then be used for FDIR design, implementation, and testing [143,146].

As shown in the [Chapter 11](#) – FDIR Development Approaches in Space Systems, the Packet Utilization Standard (PUS) [133] addresses the FDIR operational concepts in current ESA missions and, among other things, provides a set of services that allows the standardization of the fault management for ESA missions. Such FDIR PUS services encompass both on-board monitoring and recovery action capabilities, and thus allow the spacecraft to react to predefined events and subsequently select a recovery routine from a given set of options accordingly. The established FDIR principles constitute a good level of robustness for traditional satellites, are industrially mastered and established within the development process.

However, such traditional (PUS-based) FDIR systems have important limitations, especially for upcoming space missions, which demand on-board high-rich autonomous FDIR capabilities. For instance, the traditional FDIR solutions are not able to cope with the partial observability of the system. In particular, simple diagnostic (threshold-based) routines process symptoms in isolation, which may result in incorrect diagnoses or contradictory deductions [127,134,135]. One prominent example is the Mars Express mission: due to a nonresolvable memory failure, it suffered from repeated safe mode transitions, which resulted in a suspension of science operations [136].

In general, the following shortcomings of traditional (PUS-based) FDIR systems can be highlighted [134,147]:

- The PUS services dedicated to the FDIR are generally good at detecting “single” failures but limited in isolation capabilities and struggling when multiple faults combine in an even nonforeseen behavior. Even though different levels of monitoring thresholds and check types can be defined (e.g., expected-value-checking, limit-checking, and delta-checking), the overall PUS-based FDIR mechanisms process symptoms in isolations, which can result in incorrect diagnoses or contradictory deductions.
- They do not exhibit any prediction capabilities, not being able to detect early deviations from nominal behaviors of S/C components, which may result in an upcoming failure or a novelty [148].
- Another drawback of PUS-based FDIR mechanisms is the difficulty in managing failures at system level. This issue is a direct consequence of their limitation in isolation capabilities.

As per today, the above-mentioned limitations of the current FDIR solutions are partially addressed by tailoring the PUS standard. However,

despite providing the capability of monitoring on-board functions, the concept is still based on a threshold monitoring mechanism.

In order to cope with all these issues of current FDIR systems and offer new failure management capabilities to the upcoming space missions, new directions have to be explored, such as model-based and data-driven (supported by models, if needed) diagnostics solutions. Unlike current FDIR systems, they should provide the capabilities of predicting and detecting anomalies in the behavior of a satellite at unit, subsystem, and system level and isolating the related root cause. Such solutions can be combined with industrially consolidated FDIR approaches with the aim of reducing the number of safe mode events, increasing the S/C operational time, and limiting the overall operational cost.

In the field of FDIR for space missions, several studies have also been conducted utilizing both data-driven and model-based methods, see Refs. [139–141,149]. Even though these studies represent a significant step toward more robust and autonomous on-board FDIR systems for space missions, it is relevant that the only actual in-flight study in this domain was the remote agent experiment aboard Deep Space One [150]. Thus, even if the studies using model-based and data-driven methods both in the space sector and in other fields show encouraging results, there is still a lack of understanding of how these methods can be robustly integrated in on-board S/C FDIR systems. In Ref. [135], the reasons for such a widening gap between the advanced scientific FDIR methods being developed by the academic community and technological solutions demanded by the aerospace industry are discussed. Major problems can be found in the lack of an effective development process for maturing on-board implementations of advanced FDIR systems as well as verification and validation (V&V) aspects. In the remaining part of this section, we present some approaches of both model-based and data-driven FDIR systems for GNC applications, and then we propose a possible development approach for data-driven FDIR systems. Finally, the major challenges for the industrial implementation of advanced FDIR systems are discussed.

Model-based methods for implementing FDIR systems in GNC applications

In literature, as for GNC applications, the fault detection and isolation functionality of FDIR systems is often referred to as fault detection and diagnosis (FDD), while the recovery functionality can be called either fault-tolerant control (FTC) or fault-tolerant guidance (FTG), see Refs. [134,135]. FTC

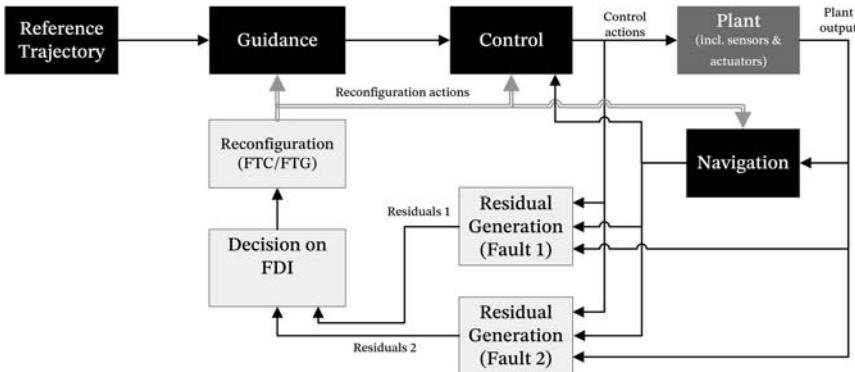


Figure 15.51 Model-based FDIR system architecture for GNC applications. Derived from I. Hwang, S. Kim, Y. Kim, C.E. Seah, A survey of fault detection, isolation, and reconfiguration methods, *IEEE Transactions on Control Systems Technology* 18 (3) (2010) 636–653.

functions aim at providing a degraded level of performance in the faulty situations, while FTG means on board reshaping of the mission objectives in case of critical faulty conditions.

The basic idea of model-based FDIR consists of using control engineering approaches for assessing residuals (fault-indicating signals) generated from the comparison of the system measurements with their estimates. Model-based FDIR solutions fall into the more general analytical redundancy approach, which, contrary to the HW redundancy, it is advantageous in terms of cost and space savings and in dealing with the inherent complexity of GNC applications [135].

Residual generation uses a model of the system in which the control actions sent to the actuators and the system/plant outputs as measured by the sensors are injected to predict the behavior of the system and compare this prediction to the actual behavior, see Fig. 15.51. This way, it is possible to calculate quantitative indices of the presence of faults. Such indices are called residuals. Different methods can be used for the design of the residual generators. For instance, robust control techniques for estimating the fault signals are an appealing approach, thanks to their ability to handle model uncertainties and disturbances [159–161]. The objective of such robust control techniques is to make the residuals sensitive to one or more faults, while at the same time making them insensitive to, e.g., uncertain disturbance effects acting upon the system being monitored [134]. Observer-based approaches are also one of the most popular among FDI design techniques, in particular, Luenberger observers for the reconstruction of the state

variables under deterministic hypotheses and Kalman filtering for the state estimation in a stochastic context [162,163].

In fault-free situations, the residuals are normally close to zero. On the other hand, they deviate from zero after the occurrence of faults to which they are sensitive. There is also the need for a residual evaluation strategy to translate the time behavior of a residual into a Boolean decision function. This usually requires the usage of thresholds or tests of statistical hypotheses. Any given residual may be sensitive to one or more faults. As a consequence, a decision logic following residual evaluation can be needed to provide additional insight into the system behavior, and thus help isolating the fault [139].

The next step following the design of an FDD system would be the definition of proper recovery/reconfiguration strategies. The general objective is firstly to ensure continuous safe operation of the system and secondly to keep some performance level in fault situations. As shown in Ref. [140], reconfiguration solutions can be classified into two main categories: multiple model and adaptive control-based approaches. As for the former, a set of parallel models is used to formulate the system under normal operating mode and under various fault conditions. A corresponding controller is designed for each of these models. Thanks to a proper switching mechanism and with the support of the FDD, the mode of the system at each time step is determined and the associated controller is activated. Another common approach in reconfigurable control is to utilize an adaptive controller approach, which means updating (or computing new) controller parameters, following the typical design paradigm of adaptive control.

Fig. 15.52 presents a classification of the various model-based FDIR techniques, which can be used in the FDIR blocks as shown in Fig. 15.51. Model-based FDIR systems are much more sensitive and much more specific than the PUS threshold approaches, e.g., early detection of anomalous behaviors, fewer false alarms, large benefit in dynamic scenarios such as station-keeping maneuvers [141,161]. Moreover, they combine failure detection and isolation and allow the design of fault-tolerant systems. However, as shown later, some issues have to be solved in order to implement model-based techniques on-board spacecraft.

Data-driven techniques for implementing FDIR systems in GNC applications

When explicit high-fidelity dynamic models are unavailable, data-driven techniques can be used to implement FDI capabilities in GNC applications.

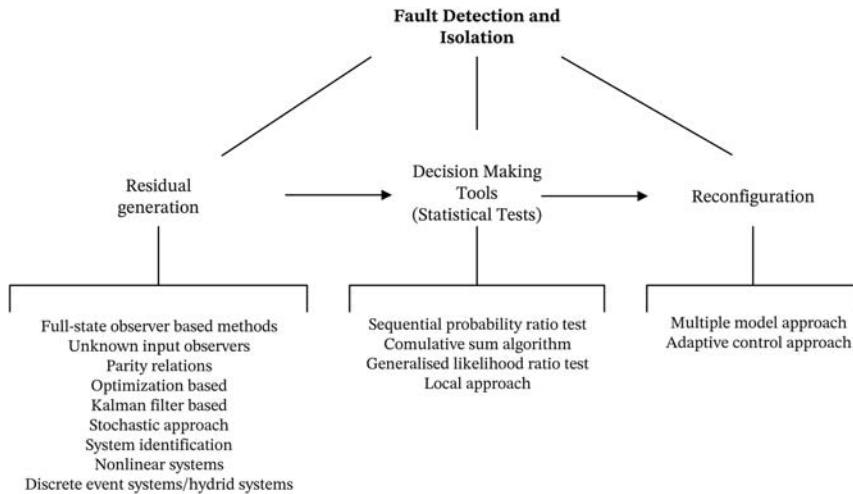


Figure 15.52 Classification of model-based FDIR techniques [140].

In other words, data-driven methodologies represent one main alternative to model-based ones for implementing FDIR systems. For instance, space-craft flight data can be exploited to build regression models (used for failure detection) and classifiers (used for failure isolation) [127,140].

AI- and ML-based approaches seem to be very suitable to implement very effective diagnostics capabilities in space applications [139,141,164]. In particular, such techniques can provide the capability of detecting incipient fault, which would enable the definition of preventive actions. As shown in Ref. [148], symptoms of incipient faults may be observed in the monitored parameters even when they evolve within their threshold bounds. Preventive maintenance can be important in view of critical mission phases, such as S/C rendezvous as well as entry, descent, and landing operations.

The aging or degradation level of a component can be assessed, and preventive maintenance can be triggered to increase the level of confidence in the success of upcoming critical mission phases. The added value of ML techniques comes from their capability of processing combinations of parameters. For instance, even though each parameter value is in its nominal range, their combination can present some anomalous conditions, which is a symptom of an aging/degradation stage of a specific component. Autoencoders and RNNs can be used for implementing such advanced anomaly detection capabilities [166]. Generally speaking, ML algorithms can deal with multiple parameters at the same time, thereby modeling their

joint behavior and patterns. Classification and pattern recognition techniques can be powerful tools for the enhancement of the fault isolation task.

As for the FDI functionality implementation, the following data-driven approaches/techniques can be adopted: pattern recognition, support vector machine, ANNs, PCA, and (dynamic) Bayesian networks [141,164]. For instance:

- ANNs do not use any a priori knowledge about the system domain. They exploit sample data to discover patterns and relationships hidden in such data as well as to extract and process relevant features applicable to the problem at hand. ANNs are mostly used for pattern recognition (in both time-series and multidimensional data such as images) and for controlling highly nonlinear systems. An interesting ANN-based FTC solution applied to a Mars entry vehicle can be found in Ref. [167].
- Bayesian networks can be used to identify the system state based on prior and likelihood beliefs in a set of system variables. They provide a means to model uncertainty and partial knowledge in the observation domain. Dynamic Bayesian networks can express dynamic behavior of a system over discrete time steps and can be used for designing and implementing highly autonomous FDIR systems [169].

As far as the recovery functionality is concerned, it should be considered that this usually requires a deep understanding of the underlying system. Expert systems can be used to implement such recovery functions [147]. Expert systems are used to emulate the decision-making ability of human experts. More specifically, knowledge-based expert systems use logical rules to guide such decision process with the support of effective searching algorithms in order to automate the exploration of the hypothesis space. Deep reinforcement learning approaches can also be used for GNC applications, in particular, for learning a policy mapping (both healthy and faulty) states to actions, see Refs. [87,168].

For more information about the usage of data-driven FDIR approaches, the reader can refer to different surveys available in the literature [149,165]. Fig. 15.53 shows a possible data-driven based architecture for FDIR systems in GNC applications and basically highlights how the different FDIR functions in GNC applications can be implemented via data-driven solutions. In particular, the FDI blocks can be implemented by using both classifiers and regressors. Classifiers perform the task of predicting a discrete class label. In particular, classes from labeled datasets can be built and used to train such classification models, which are then adopted to assign newly (unseen) operational conditions to classes representative of healthy or faulty behaviors. On

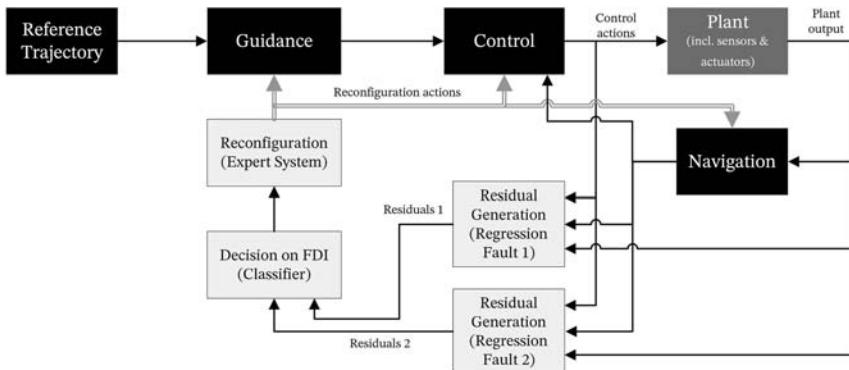


Figure 15.53 Data-driven system FDIR architecture for GNC applications. Derived from J. Marzat, H. Piet-Lahanier, F. Damongeot, E. Walter, Model-based fault diagnosis for aerospace systems: a survey, *Journal of Aerospace Engineering* 226 (10) (2012) 1329–1360; I. Hwang, S. Kim, Y. Kim, C.E. Seah, A survey of fault detection, isolation, and reconfiguration methods, *IEEE Transactions on Control Systems Technology* 18 (3) (2010) 636–653.

the other hand, regression is the task of predicting a continuous quantity. Regressors are statistical models (trained on the process data) and can be used for the prediction of upcoming failure conditions [139,148]: they can use redundancy in the process history to predict the values of variables and generate residuals by comparing predictions to measured values. The recovery function can be implemented via expert systems.

Development workflow for data-driven FDIR systems

In this paragraph, we show a possible development workflow for data-driven/ML-based FDIR modules in space systems, including GNC applications. The presented workflow is designed in a way to guarantee the full harmonization with traditional FDIR system development processes [127,138].

Fig. 15.54 shows the proposed workflow for developing ML-based FDIR solutions and integrating them into space systems, considering the current FDIR system development process and the need of establishing the above-mentioned data engineering approach. The overall workflow, named ML-based FDIR workflow, can be based on the union of four subworkflows:

- ML development subworkflow (step #1, #2, and #3 of Fig. 15.54): Starting from the raw data, such subworkflow generates as output a trained ML-based FDIR module, incorporating FDIR functionalities tailored to, e.g., the specific GNC application.

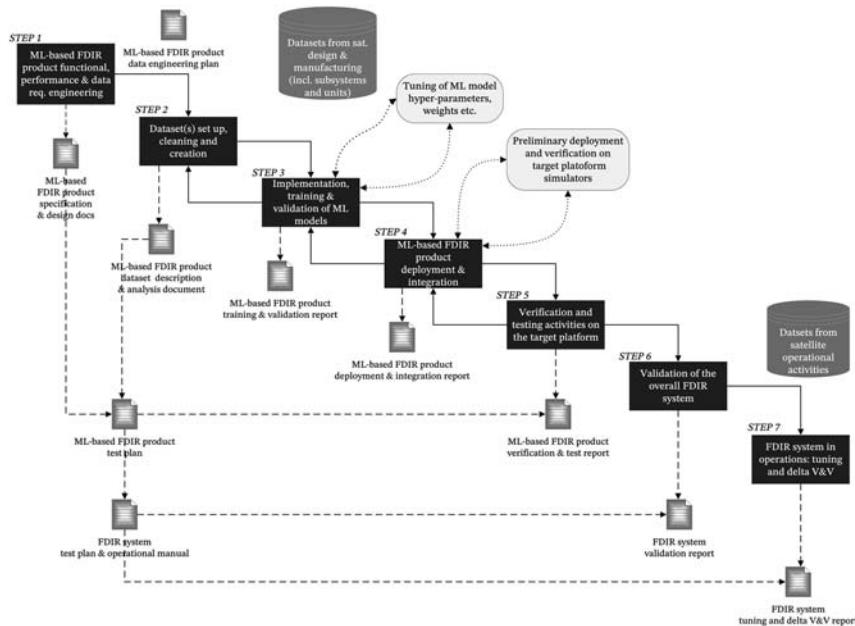


Figure 15.54 Development workflow for ML-based FDIR systems.

- ML deployment workflow (step #3 and #4 of Fig. 15.54): Starting from the trained ML-based FDIR module, such subworkflow optimizes it and generates an SW module, named deployed module, suitable to run in embedded platforms.
- ML testing subworkflow (step #5 and #6 of Fig. 15.54): Starting from the deployed ML-based FDIR module, such subworkflow verifies and validates the module on the target platform, after being integrated into the hosting on-board SW (OBSW). This workflow generates the verified and validated ML-based FDIR module.
- ML operational subworkflow (step #7 of Fig. 15.54): The verified and validated ML-based FDIR module is used during the actual S/C operational phase. Flight data instances can be collected and used to enhance the FDIR products.

ML-based FDIR approaches rely on data. This means that the same ML model architecture can be applied to capture system(s)/subsystem(s) behavior on different missions, provided that it is trained with the specific representative mission data. Therefore, an ML-based workflow can be more easily applied and iterated to develop FDIR solutions to address similar

use cases on different missions. This would enable a higher generalization of the workflow and scalability of ML-based FDIR solutions.

ML-based FDIR solutions may not be as interpretable as traditional approaches based on thresholds, especially when DL is applied. However, such solutions have the advantage that the input data used for training and testing the models/algorithms are known. This means that it is possible to characterize the performance of the model and its validity domain.

Finally, it is worth highlighting that the achieved performance depends on the data availability and their representativeness of the faulty conditions in GNC applications. Data from assembly integration and test (AIT) activities and from the real spacecraft operations can be used. If high-fidelity simulators are available [158], additional synthetic datasets can be produced and adopted for training, validating, and testing ML-based FDIR algorithms.

Challenges and next steps for the industrial implementation of advanced FDIR systems

Two major obstacles can be identified for the implementation of advanced FDIR modules in space systems, including GNC applications: the definition of a proper V&V process and the deployment of model-based/data-driven FDIR solutions on space qualified computers.

The V&V process definition of both model-based and data-driven FDIR systems is a topic of paramount importance toward the adoption of such algorithms in space missions, where safety and reliability aspects are strongly considered [127,147]. Traditional V&V techniques [151,152] do not scale to on-board high-rich autonomous systems, since optimization and model-based algorithms are at their core [153]. More specifically, the inadequacy of traditional V&V techniques is due to the fact that advanced FDIR systems are made up of AI/ML algorithms and models describing the domain on which the reasoning is performed and the engine/framework for such models to work [154]. Such algorithms and models tend to be very sensitive to the environment, and one should explore the behavior of the system over a vast range of plausible conditions in order to demonstrate their robustness. As a consequence, it becomes necessary to employ more advanced and emerging V&V techniques, which are largely based on analytical methods.

The three main approaches toward the deployment of reliable advanced FDIR systems in space missions are [147]:

- *Formal verification.* The concept of mathematically proving that an algorithm meets its specification [155,156].

- *Runtime verification.* The concept of monitoring at run-time the execution of, e.g., AI/ML algorithms.
- *Iterative approach.* The concept of accepting the deployment of an algorithm verified with traditional methodologies, and then tuning confidence thresholds (e.g., thresholds for classification problems) over time, as more knowledge is gained in the operational environment.

The latter approach recalls a V&V philosophy in which the examination of the FDIR system is performed up to its acceptance review (with acceptable open risks). Afterward, further periodic V&V examinations can be carried out during the actual operational phase by exploiting flight data extracted from the S/C telemetry streams.

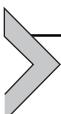
Both model-based and data-driven FDIR solutions must be integrated into the OBSW and deployed into S/C on-board computers. In this regard, two constraints must be taken into account:

- The ECSS SW requirements for the criticality category B must be fulfilled [152,158].
- Space-grade processor modules are featured by limited processing power and memory resources [137].

As for terrestrial applications, and in case of ML models, there are different ways of performing the conversion of such models into deployable SW source code. For instance, one can use an automatic code generator provided by many commercial tools to convert a trained neural network into source code, ready to be compiled and linked to the host application [157].

The development of advanced FDIR systems in space projects (including GNC applications) has lagged terrestrial applications for several reasons: space-qualified computers have significantly less processing power than their terrestrial equivalents, while space reliability requirements are very stringent. As for ML models, powerful frameworks now exist to design and train network models as well as automated tools to deploy trained inference models on a range of HW targets. However, these tools are aimed at terrestrial HW devices, which are significantly more powerful than the current generation of radiation hardened space processors, e.g., LEON processor family, meaning that these tools are not able to target these processors [137]. In other words, even if the usage of automatic code generator tools for space projects is important to foster advanced FDIR applications on-board, such tools must fulfill the ECSS SW standard requirements and take into account the limitation of the space-grade processor modules (central processing unit [CPU] and memory).

The analysis reported in Ref. [157] of the available automatic code generator tools for ML models shows that even if they are suitable for terrestrial application, they have some relevant issues for their application to space projects. For instance, as for AI/ML algorithm coding, TensorFlow-Lite has a library overhead not compatible with the memory capabilities of current space processor modules, while MatLab C Coder generates C++ code, yet it has significant runtime library dependencies of a size comparable to TensorFlow-Lite. An interesting solution is offered by the TFMin library that has been recently developed at the Surrey Space Center, Surrey University. As shown in Ref. [157], TFMin supports the currently used space-grade processors and allows converting a TensorFlow graph within a Python script into a C++ implementation with only standard library dependencies. This conversion is done verbatim, so the C++ version is mathematically equivalent to the TensorFlow. Unlike the standard C++ implementation of TensorFlow (which is also available), the binaries produced by TFMin do not have dependencies on large-shared object libraries. TensorFlow operations are converted into C++ by using a dictionary, which can be extended if necessary. TFMin also computes the memory and CPU metrics of the deployable SW modules, thus allowing the quick analysis of different network topologies and the effects of changes to, e.g., the instruction and data cache sizes.



Small satellites/CubeSats

This section provides a practical outlook on GNC and ADCS subsystems design for nanosatellites. The content is tailored to designers coming from academia or larger spacecraft. The topics described here represent a particular declination of the general ones described in Part I and Part II of the book.

Introduction

The process of miniaturization in the space industry started in the 70s and soon demonstrated the capability of obtaining satellites able to achieve the same scientific and technological objectives while reducing mass and cost at launch. Satellites weighing tons of kilograms were gradually substituted with S/C below 500 kg (small satellites). In the following decades, this trend strengthened, confirming the potentiality of very small platforms up to the cm-scale in space exploration.

Table 15.5 Small satellites classification.

Category	Mass range
Small satellite	500–180 kg
Minisatellite	180–100 kg
Microsatellite	100–10 kg
Nanosatellite	10–1 kg
Picosatellite	1–0.1 kg
Femtosatellite	0.09–0.01 kg

Table 15.5 summarizes the clear classification provided by the Nasa Small Spacecraft Technology Program [170] of the various platforms depending on their wet mass at launch.

Just in 2019, almost 400 satellites with mass below 600 kg have been launched, making up 11% of all the mass launched in orbit that year [171]. However, it is important to mention that more than 60% of all the small satellites launched since 2012 are CubeSats.

The CubeSats are a particular category of small satellites, made up with multiple cubic modules of about 10 cm side. They were initially developed in 1999 by Stanford University and California Polytechnic State University to perform technology demonstration and scientific research in LEOs. The idea was to create a simple platform, similar to Sputnik, to be used by graduate students and able to host major subsystems such as batteries, solar panels, and communication systems [172].

The CubeSat platform was quickly adopted by main institutions around the world: other universities, companies, and governments started exploiting this design because it was flexible enough to cover different mission objectives while maintaining a common architecture. In fact, the key to the success of these satellites lies in the standardization of the design: precise specifications are available for the designers to create a CubeSat which is compatible with most of the deployers. **Table 15.6** provides an overview of the different configurations mostly used nowadays, together with some figures useful in the design of the ADCS of the CubeSats.

Table 15.6 CubeSat standard form factors.

Form factor	Envelope	Mass	Max inertia (at deployment)
1U	100 × 100 × 100 mm	1.33 kg	0.002 kg m ²
3U	100 × 100 × 340 mm	4 kg	0.042 kg m ²
6U XL	226 × 100 × 366 mm	12 kg	0.185 kg m ²
12U XL	226 × 226 × 366 mm	25 kg	0.385 kg m ²

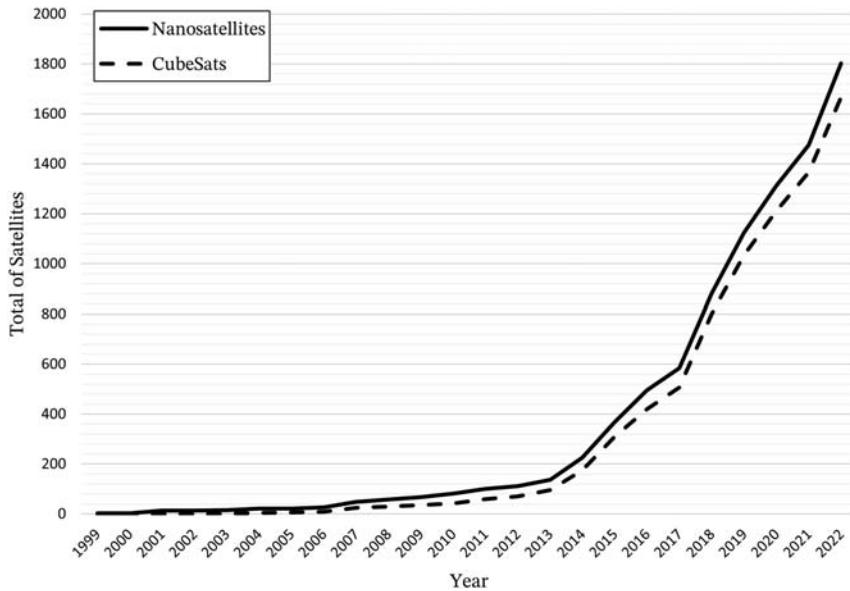


Figure 15.55 Trend of launched CubeSats.

Since the first launch in 2003, the number of CubeSats launched per year has exponentially grown, leading to a total of more than 1550 satellites by mid-2021 (see Fig. 15.55 from Ref. [173]).

Most of the launched CubeSats have a 3U form factor and are employed for Earth observation and technology demonstration. However, these platforms are proving to be flexible and reliable, while still being cost-effective and for this reason the trend is expected to be confirmed for the following years.

Hardware limitations

The main difference which we encounter HW-wise on nanosatellites, with respect to their larger counterparts, is the widespread use of electronic components or even modules/subsystems not originally rated for space use, but rather for lower standards, such as industrial, automotive, and only sometimes military or aerospace. This is a double-edged sword: on one hand, it opens up to an immense variety of choice and miniaturized technologies, while on the other hand, the performance of these devices in the space environment is hard to predict and often time-limited.

This section describes the main limitations connected to CubeSat HW, from miniaturization to size and mass, pointing performance, and thrusters.

The burden of miniaturization

One of the main causes of commercial off-the-shelf (COTS) components failure in nanosatellites is radiation susceptibility; thermal issues due to tighter operative ranges or cycling can be an important factor, but they are easy and overall cheap to predict, test, and mitigate—the same cannot be said about radiation; more and more emphasis is being put on this aspect recently, especially in the view of interplanetary CubeSat missions [170]. It must be said that this compromise of flying nonrad-hard or in any case nonspace-rated components has been precisely what enabled the CubeSat and nanosatellite field, in general, to come into being and flourish as it did. It would be hard to imagine a faster “miniaturization revolution” in the space industry which would not thread this path of commercial, non-qualified electronics.

As of 2022, approaching almost 20 years since the first CubeSat launch, a significant amount of data has been amassed on the in-orbit behavior of several of these components and new subsystems, some of which can now be considered de-facto space-qualified by virtue of their flight heritage. Therefore, the issue of flying new components is now quite reduced even in the CubeSat world — although it’s not always easy to recover information on qualification status or flight heritage for every item.

Even for components with flight heritage, and due primarily to cost and lead time constraints, nanosatellite manufacturers will very rarely procure rad-hard options, rather looking into rad-tolerant architectures or just accepting the relative risk. This often results in spacecraft having a relatively limited lifetime (in the order of months to a few years) or more generally to experience performance degradation in orbit, as some subsystems may accrue damage or experience, in time, uncorrectable faults.

Size and mass limitation

Another factor which contributes to the limited lifetime and/or performance of nanosatellite systems is certainly the limited mass and volume budget, even taking into account the advantages brought by miniaturization.

A first handicap brought by these constraints is the unsuitability of higher performance or ruggedized components due to their excessive bulk. Sometimes, the state-of-the-art technology for a certain device simply is not or cannot be miniaturized enough for a sensible nanosatellite application; for instance, ring laser gyros (RLGs) are not yet available in CubeSat-friendly formats, and therefore designers usually need to make do with much less-performant microelectromechanical system (MEMS) rate sensors. Other

times, manufacturing quality simply can't scale well enough with size: a clear example being represented by reaction wheels. These need very high-quality bearings and ultraprecise balancing; the smaller the wheel, the higher the impact of the absolute machining and balancing tolerance limits reachable by current technology, and therefore the lower the attainable quality. Furthermore, smaller wheels require higher rotational speeds, which in turn accelerate wear and tear of the component. Finally, the smaller the component, the higher its sensitivity to defects, damages, and contamination, which can accrue during manufacturing, AIT, launch, and in-orbit operation.

A second matter to deal with is the concept of redundancy. While CubeSats can and do employ varying degrees of redundancy at component level, it is quite rare to see significant redundancies at module level and even less at subsystem level. In general, the trade-off needle tends to swing more toward lower mass/volume and few fully redundant items. Still, among all the subsystems, ADCS can often enjoy a certain degree of resilience, with multiple sensors and actuators enabling at least a “graceful degradation” if any single fault is encountered. Nevertheless, important compromises need to be taken in selecting an architecture, in particular, for what concerns star tracker heads and reaction wheels.

Having multiple star tracker heads increases sky coverage and the likelihood that at least one head has a clear view of the starfield, unobtruded by the Earth or solar glare. On the other hand, star tracker heads are among the bulkiest modules in a CubeSat, especially when accounting for their baffles. While larger platforms with 3+ heads are not unusual, most nanosatellites shipping star trackers will have at most two heads, and often just a single one.

Reaction wheels (cfr. [Chapter 7](#) – Actuators) can traditionally be mounted in four wheels tetrahedron or three wheels orthogonal configurations. The former still allows full three-axis control if one wheel fails and furthermore allows angular momentum redistribution if all four wheels are operational. However, this arrangement can be geometrically awkward, the canted wheels reduce the packing efficiency of the spacecraft and require a larger volume. Most of the time, this forces the designers to employ smaller wheels than could otherwise be used in the orthogonal configuration. The trade-off is therefore often played on the emphasis given to the robustness and quality of the single wheel versus the resiliency of the whole subsystem. Both solutions are well represented among commercial platforms.

Finally, it is worth noticing that an additional nonengineering factor can further exacerbate the described issues: system cost. CubeSats have been

brought forward as low-cost and high-risk platforms to broaden the access to space, and therefore often these programs run on very tight budgets; this can easily prevent the employment of highest tier components or redundancy even in the rare instances in which the envelope constraints would allow it.

Pointing performances

Nanosatellites are often employed in LEO missions which are very dynamic with respect to attitude control. Most CubeSats need to perform Sun-pointing phases to maximize their power input, ground tracking during ground stations passes to enable the use of directive antennas and improve link budget, local-vertical-local-horizontal (LVLH)-fixed or Earth-centered inertial (ECI)-fixed attitudes depending on their payload and mission requirements, and so on, sometimes all in a single orbit. For instance, this is very much different from a GEO spacecraft, which can mostly adopt a single attitude for the whole duration of its operational lifetime, and for which sensor placement can be well optimized by design.

The large field of view taken by the Earth limb in LEO, the extremely varied attitudes, and the limited number of sensors available in a CubeSat (especially star tracker heads), all conspire against the consistency of attitude knowledge and, consequently, pointing performance. Depending on the vehicle attitudes, HW configurations, and ambient conditions, the actual performance can be degraded even by two orders of magnitude or more when conditions are unfavorable, with respect to the ideal case.

Therefore, in missions requiring high pointing precision, matching the specifications of the attitude determination sensors and filters response to the requirements is not sufficient to guarantee adequate performance in orbit. The chance and impact of not having consistent fixes due to Earth or Sun occlusion or glare need to be carefully evaluated during the design and verification phase, at the very least computing pointing performances for daylight and eclipse, and better yet with a full sensor access/visibility analysis, taking into account the actual orbit and mission attitudes. Note that the lighting conditions in LEO are time-changing also for Sun-synchronous orbits (SSOs), so these analyses shall not be restricted to an individual period, but possibly to be repeated for different times of the year.

Thrusters

The field of nanosatellite propulsion has enjoyed in recent years an outstanding growth and diversification of available solutions, especially in the field of electrical propulsion. Mainly due to the favorable power to

mass ratio of CubeSats, niche technologies such as field-emission electric propulsion (FEEP) or radio-frequency thruster (RFT), traditionally unsuitable for main propulsion in larger spacecraft, are finding fertile ground for application on nanosatellites [170].

While early CubeSats didn't usually employ propulsion, and still only a minority of these spacecraft ship thrusters, the need for this subsystem has been growing lately, driven in particular by the requirements of commercial constellations deployment (rephasing, orbit maintenance, etc.) or proximity operations missions.

Installing a thruster module in a nanosatellite poses a series of challenges to the ADCS and interfaces design which need not be underestimated:

- The most important criticality is given by parasitic torques. Many propulsion modules for CubeSats, especially in the case of electric propulsion (EP), only offer axial main thrust, with no gimbaling or thrust vectoring ability. This means that any misalignment of the thrust axis with the center of mass will induce parasitic torques on the spacecraft. These torques must be absorbed and compensated by the ADCS actuators, e.g., instantaneously by reaction wheels and long-term (possibly considering also thruster duty cycling) by magnetorquers. Note that due to the typical mechanical workmanship tolerances in CubeSats, it's not unusual to have misalignments in the range of 0.1 mm and of 0.1 degrees. To these one must add the uncertainty in the center of mass position and in the actual thrust axis with respect to the thruster reference axis. For cold or warm gas systems, in which the generated thrusts are greater, the effect can be even greater, and unmitigated parasitic torques can reach values in the mNm range, which can be quite challenging to compensate.
- Throttleability is not a typical feature of nanosatellite propulsion systems, for both EP and chemical propulsion. Control algorithms are best designed assuming no continuous thrust control, but rather ON-OFF or at best pulse-width modulation (PWM) if the module supports it. Moreover, many systems require preheating/preconditioning phases before a firing or might only support firings of a certain maximum duration due to thermal or other design constraints. These limitations shall be explored when selecting a thruster and considered in its control logic.
- Many propulsion modules designed for CubeSats can only be mounted in specific orientations and accommodations, which can prevent their application on some form factors. For instance, there's a large number

of modules which exploit the “tuna can” volumes on the -Z faces — this is basically an additional volume which protrudes from the basic CubeSat shape and fits within the dispenser pusher plate springs [176]. While this is an excellent way to save volume, it is mostly practical for 3U vehicles, since the tuna can volumes are not centered on the Z axis for 6U or 12U form factors. While mounting multiple modules in parallel can be a viable option, thrust imbalances, misalignments, and synchronization among the various modules must be considered.

COTS components

One of the novel approaches brought to the space industry by CubeSats and nanosatellites, in general, is the large use of COTS components, often also coming from different industrial sectors.

The defining feature of a COTS item is that it's procured “as is,” and not tailored to the specific application. If coming from a different industrial sector entirely, it could even not be designed for use in space, with all the performance and reliability implications which this can entail. However, this doesn't mean at all that these components can't be flight proven, i.e., at technology readiness level (TRL) 9. There is plenty of COTS not originally designed for space which have now flown on several CubeSat missions, even to the point of becoming a de-facto standard [170]. This section will strive to provide a set of guidelines to the aspiring CubeSat manufacturer, component by component, which are equally valuable in both selecting a COTS component for their application, or in drafting up the specifications for a custom development.

COTS or custom?

The choice between using COTS or developing and qualifying a custom subsystem naturally depends on a wide array of factors, such as budget, schedule, know-how availability, scale of the required production, product TRL, and so on. Note that most of these factors are purely programmatic, although there certainly are cases in which this trade-off is driven by the unavailability of COTS which meet the required performances; this tends to occur less and less as the subsystems market matures and is more frequent in more niche or varied applications such as propulsion.

Due to these constraints, universities and very small companies will tend to rely on readily available (and potentially flight proven) subsystems, focusing instead on integrating them into a single platform. Larger companies, as they grow, will usually be able to spend some research and

development resources into developing their own custom avionics and standard platform, optimizing their design in order to be able to accommodate a wide array of payloads. This allows to reduce as much as possible both the nonrecurrent engineering and the recurrent production costs for all the subsequent missions, besides granting a higher level of control on the supply chain and the production schedule. Beyond the initial investment, and for large enough production volumes, the “make versus buy” trade-off inevitably tends more toward the former.

There are, however, several exceptions to these generic remarks, especially for ADCS/GNC subsystems. Some components, due to the high level of background technology and know-how involved, are bought as COTS by most all CubeSat integrators—this includes, for instance, GPS receivers or IMUs. On the other hand, some other components such as magnetorquers are easily manufacturable with optimized specifications by any integrator.

Magnetometers

Magnetometers are among the most widespread sensors used in CubeSats. The wide availability of cheap and compact IC-size COTS, coming from all kinds of industrial sectors, makes it especially easy to integrate these sensors into a CubeSat.

Moreover, the information they provide, on the local magnetic field, is the basis for both the simplest and crudest attitude control systems (pure magnetic pointing with magnetorquers), as well as the more sophisticated ADCS, in order to effectively implement magnetic desaturation techniques and augment the attitude determination filters.

Several technologies exist, from the cheapest but noisiest magnetoresistive sensors up to the higher performing magnetoinductive and fluxgate magnetometers.

Regardless of the specific technology, the key design parameters to look for in a magnetometer are quite standard:

- Sensitivity, which determines how precise a sensor can be and therefore the minimum achievable angular resolution of the measurements.
- Noise characteristics, which includes intrinsic noise floor, nonlinearities, stability, orthogonality, temperature susceptibility, etc. All of these metrics impact on the quality of the measurement and can limit both its accuracy and precision. Besides the information reported in the datasheet, there is no substitute for a good noise characterization campaign in a representative installation.

Range is also important, i.e., the sensor needs to not saturate in the Earth magnetic field ($+/- 1\text{G}$ is enough); however, the vast majority of magnetometers suitable for spacecraft use are already designed with geomagnetic sensing as a target application and are thus typically compliant with this requirement.

Besides the intrinsic characteristics and ideal performance, however, a key issue with most magnetometers is the environmental noise which derives by their application within a spacecraft. It's no coincidence that scientific magnetometers on interplanetary probes are mounted on very long booms to keep them away from the main bus. Spacecraft, and especially small- and tight-packed ones (such as CubeSats), are very noisy environments, due to a plethora of effects, such as hard and soft iron magnetization of the structure and components, or the fields induced by power buses, current loops, and electronic devices.

The following are examples of noise sources which generate a field of 0.1 G (thus on the same order of magnitude as the geomagnetic field in LEO):

- A wire carrying 1 A current at 2 cm distance.
- A single 0.1 Am^2 air-core magnetorquer with a radius of 5 cm , $\sim 12\text{ cm}$ down its axis.
- A single 1 Am^2 iron-core rod magnetorquer at $\sim 20\text{ cm}$ distance, along its boresight.

All of these are totally plausible configurations which can be encountered in a CubeSat.

Not only these sources of noise can be significant but also they are challenging to filter, having components which are attitude-dependent, time-dependent, and generally hard to predict.

The set of strategies which can be employed to limit or mitigate this form of noise is known as magnetic cleanliness; in particular, good practices on power harnesses (e.g., twisted pairs, low-dipole layout of solar arrays, etc.) is of paramount importance in CubeSats.

Sun sensors and earth sensors

These classes of sensors are able to detect either the Sun direction or the Earth limb to a precision typically in the order of 1 degree. They are employed as coarse attitude sensors, either as backup, or sometimes even as cheap alternatives, with respect to finer sensors such as star trackers.

There is a wide variety of options available for the CubeSat market, both in terms of individual products, of performance, and of technologies employed.

- Cosine detectors are very simple devices, which estimate the solar incidence angle based on the cosine-like response of a photocell. More such sensors on differently oriented surfaces of the spacecraft are required to fully reconstruct the solar direction angle. Sometimes the power-generating PV cells themselves can be used to reconstruct solar direction information with this technique.
- Position-sensitive devices/quadrant detectors instead provide a full reconstruction of the solar angle, by either shining the sunlight through a pinhole or slit over an isotropic detector, or by employing special anisotropic detectors.
- Visible/infrared cameras are based on proper imagers, with a set of optics and an imaging detector. They can be used both for Sun and Earth detection. Infrared horizon sensors have the advantage of working both in sunlight and eclipse.
- Horizon-crossing indicators only provide, as the name tells, a binary information which changes when the Earth limb crosses their field of view. They are typically based on pyroelectric sensors or microbolometers.

Depending on the selected technology and complexity of the sensor, there is a significant span along the performance versus size/power/cost spectrum, from quite coarse >1 deg resolution IC-size detectors up to proper cameras offering 0.1 deg accuracy.

While a good Earth sensor might be an excellent choice, especially for missions requiring prolonged nadir pointing, the great advantage of using Sun sensors is that, in contingency situations, they can provide the primary information required to expose the solar arrays to sunlight and therefore keep the vehicle charged.

Typically, for this application as a backup or complementary sensor, even a few degrees of precision are enough (even five degrees of pointing error only result in <1% of cosine losses on the solar arrays)—it is instead more important to be able to support a sufficient number of them on the various faces of the vehicle, so as to possibly always have the knowledge of where the Sun is, no matter the attitude.

In ingesting the data provided by these sensors, one should pay close attention to the effect of the Earth albedo, which in LEO can reach a very significant fraction (same order of magnitude) of the direct solar radiation.

Another important aspect to consider is the field of view of the detector, in order to have as much coverage as possible with the fewest sensors. For multisensor applications, ideally the FOV needs to be wide enough to merge with the ones mounted on different faces, in order not to create “blind spots.”

Star trackers

Among the subsystems covered in this section, star trackers are certainly among the most sophisticated and expensive. They are, however, basically the only available sensor option which enables subdegree pointing accuracy. Good quality star trackers can provide attitude knowledge down to a few arcseconds, even on CubeSats.

Star tracker design is dominated by delicate technical trade-offs; the objective is to have a compact visible imager which can consistently and accurately image, detect, and identify stars in its field of view, in a sufficient number to generate an attitude estimation.

Technological constraints on the detector sensitivity and noise characteristic, practical requirements on the exposure time, and the minimum star magnitude which needs to be captured, all play together to impose a lower bound on the minimum geometrical aperture that the optics can have. While this limit is certainly fuzzy, it’s rare to see star trackers with an aperture much smaller than a couple of centimeters. While this is not particularly large even for a CubeSat, one needs to also consider the size of the baffle, which is of paramount importance to cut out sunlight and earthshine stray light, and often has a volume comparable, if not greater, to the star camera itself. In the end, there’s a definite trade-off between the volume dedicated to these sensors and their performance, in particular, in terms of exclusion angles.

The key performance metrics of a star tracker are:

- Accuracy that should be expressed in arcseconds at least across-boresight and about-boresight (the two figures are typically even one order of magnitude apart).
- Field of view, which goes hand in hand with sensitivity (minimum star magnitude detectable). These two parameters dictate the probability of having enough stars in a single frame to generate a solution. One of the solutions to the size issue of star trackers could be to limit the search to the brightest stars, and thus ship a smaller optics; however, this requires a larger FOV and increases the probability of having the Sun or the Earth in it. For this reason, most manufacturers settle on FOVs in the 10–20 degrees range [170], and image stars down to the sixth magnitude.

- Exclusion angles with the Sun, Moon, and Earth limb. This is the minimum angle from the boresight that these objects must have in order not to introduce excessive stray light (and thus make an attitude solution impossible). This is mostly a function of the baffle geometry, and there's a definite trade-off between smaller baffles and better exclusion angles.

On the matter of exclusion angles, this fits into the broader topic of star tracker access, i.e., the amount of time over an orbit in which the star trackers can achieve a solution. The GNC designer shall do well to perform a visibility analysis, taking into account the evolution of the mission attitudes, the placement of the star trackers on the vehicle, and the exclusion angles, to determine whether the star trackers will be able to perform as expected, and how long the “blackout” periods in which no solution is possible might be. Note that while the Sun usually mandates the largest exclusion angle, it's still basically a point source; CubeSats are most often employed in LEO, where the Earth limb will carve away more than 30% of the celestial sphere at any time, even before considering any exclusion angle. So, star tracker access might be even more sensitive to Earth exclusion angles than to the Sun ones.

Star trackers can also be quite sensitive to temperature—while the electronics may be rated to operative temperatures well in the 80°C range, the exponential increase in thermal noise on the detector can impair its ability to detect faint stars at much lower temperature. It's always important to always obtain from the manufacturer the operative temperature range at which the star tracker can still achieve attitude fixes.

Inertial sensors

CubeSats ship a wide array of inertial sensors, from pure gyro packages to full-fledged 6-DOF IMUs. Due to the very specialized nature of these components, they're almost invariably procured as COTS. Focusing on gyroscopes, which are by far the most used sensors, there are two main technologies available: fiber-optic gyros and MEMSs — the former offering higher performance, the latter higher miniaturization and lower cost. The miniaturization of even more sophisticated technologies, such as RLGs, is unfortunately not advanced enough to allow their application on CubeSats [174].

Star trackers can have blackout periods due to exclusion angle violations, and, in general, provide lower frequency attitude information. It can very well be, therefore, that a CubeSat attitude knowledge performance is not really limited by the star tracker, but by the IMU/Gyro package. An

arcsecond-level ST fix at 1 Hz, with occasional minute-long blackouts, won't do much good if the gyroscopes precision and bias stability can't maintain that level of accuracy in between fixes.

The key design parameters to look for in an IMU are (cfr. [Chapter 6 – Sensors](#)):

- *Random walk noise*: It is strictly related to the single-measurement precision of the device and is the limiting factor on shorter timescales (e.g., in-between consecutive ST fixes).
- *Bias instability*: It is related with the amount the intrinsic sensor bias drifts over a given period; this ultimately limits the accuracy of the measurement over longer timescales, whenever no external measurements can contribute to estimate the bias (e.g., during an ST blackout).
- *Temperature sensitivity*: It is related with the amount the sensor performance figures degrade or vary with temperature. This effect can be particularly severe for MEMS devices and must be taken into account for CubeSat applications, in which the LEO sunlight/eclipse cycle is very likely to induce appreciable cyclical temperature fluctuations in the avionics.

On the matter of accelerometers, these are naturally only really required in case of on-board propulsion. A good sensitivity is required to measure the tiny accelerations imparted by CubeSat propulsion systems. Warm and cold gas systems will generate accelerations in the 10^{-5} to 10^{-3} g range, which may or may not be detectable with sufficient resolution by a MEMS accelerometer, depending on the model, while EP systems, generating down to 10^{-6} or 10^{-7} g will most likely be undetectable by direct measurement, and indirect methods on longer timescales, e.g., from GNSS position fixes, might be required to close the loop.

A final word must be spent on potential issues stemming from helium susceptibility in MEMS [[175,176](#)]. Some MEMS devices can be either temporarily disabled or have their biases disproportionately amplified beyond standard specs, if exposed to sufficient concentrations of helium. It is believed that the helium, given its very small atomic radius, is able to diffuse into the device cavities (which are usually under vacuum) and interfere with the MEMS oscillator movement [[175,176](#)]. A CubeSat might be subject to an appreciable concentration of helium when inside the launch vehicle fairing, as some launch campaigns require helium purges or pressurizations of the fairing volume. Fortunately, the effect is usually reversible once the vehicle is exposed to vacuum, though over a timescale which can be in the order of days.

GNSS receivers

This is yet another class of very specialized items for which procuring COTS receivers is almost a must. Fortunately, power and size are now miniaturized enough for the available industrial solutions that the technology is already “CubeSat-ready” in terms of these specs. There are a few widely adopted solutions in the market which have accumulated very significant space heritage.

A trade-off which the designer might have to face is the selection of which GNSS systems and bands to support, spanning from just GPS L1 up to the whole suite of GPS, GLONASS, Galileo, BeiDou, etc. This is basically a trade-off between component cost and availability of visible satellites in orbit. With a good antenna, even the lower end of this spectrum typically provides more than enough visibility in orbit to get consistent fixes.

Note that GPS receivers are subject to the COCOM limits, which normally prevent their use above speeds of 600 m/s [177]; this would result in a very unpleasant surprise, manifesting itself only once in orbit. It’s therefore of paramount importance to apply for the proper licensing which allows to buy and use receivers without COCOM limitation.

Finally, an extra practical recommendation is to pair the GNSS receiver with a high-quality GNSS antenna (they come with an integrated low-noise amplifier) and to mount said antenna on a good ground plane, with good visibility to space considering the mission attitudes; remember that GNSS signals are very low intensity signals, practically buried in the noise floor, and any receiver system is therefore extra susceptible to electromagnetic interference issues or link budget degradation.

An argument could be made on the need, in general, for a GNSS receiver on a CubeSat, given the availability of TLEs. These constitute another good quality absolute position information source, and furthermore they are free. The error given by TLEs is typically well below 10^3 m, and although this is one to two orders of magnitude worse than that provided by a GNSS receiver, the APE component it introduces in pointing tasks won’t usually impact more than 0.1 deg for ground tracking and even less for LVLH attitudes. The RPE components will similarly be quite small, given that the TLE error is slow varying along an orbit. Except for missions with propulsion, or particularly stringent pointing or position knowledge requirements, TLEs will therefore provide sufficient precision.

- The use of TLEs, however, comes with two quite significant caveats:
- The first, and most obvious, is the need to constantly update this knowledge on-board through uplinks and to propagate on-board the last known TLE via a SGP4 propagator. If the TLEs are not updated from ground, the position knowledge will inevitably drift.

- The second issue pertains to LEOP. Rideshare launches are becoming more and more prevalent, with tens if not hundreds of different vehicles deployed on the same orbit, and they're typically a cheap launch opportunity for CubeSats. However, this makes it harder and harder to distinguish just which TLE is the one belonging to one's CubeSat among all the objects released in that particular launch. Without an on-board GNSS receiver to provide an independent fix against which to verify all published TLEs, all one can do is wait for the natural separation and follow a tedious process of elimination, for instance, by tracking the different objects with ground stations and checking which TLE best matches the Doppler shift of the RF signal. Until then, one can expect to get very poor RF links with the spacecraft.

These reasons alone might well induce the designer to include a GNSS receiver even when it wouldn't be strictly required by position knowledge requirements.

Reaction wheels

The technology behind reaction wheels is deceptively plain: they are not much more than brushless direct current (BLDC) motors with a high inertia rotor, controlled either in speed or torque (or both). However, even in larger spacecraft, reaction wheels are very delicate components, which present a large variety of possible failure modes, and require particular care in their design, manufacturing, testing, and handling. Most of the issues associated with reaction wheels are due to the fact that they necessarily host moving parts—and specifically, fast-moving parts. In a CubeSat, the reaction wheels will very easily be the only moving parts in the vehicle, besides one-shot deployment mechanisms.

Not all CubeSat missions require reaction wheels, though virtually all modern commercial platforms use them. If they are required, it is very highly recommended to select high TRL, robust, and well-qualified reaction wheels.

The main design parameters when selecting a set of reaction wheels are just two (cfr. [Chapter 7 — Actuators](#)):

- *Momentum capacity*. It fundamentally drives the maximum slew rate attainable by the spacecraft and the maximum storable momentum induced from external torques before saturation (and mandatory offloading).

- *Peak torque.* It drives the maximum angular acceleration and must be considered in combination with thrusters and other actuators, as well as control requirements, to guarantee sufficient control authority.

For most missions, the momentum capacity will be the most critical figure. Disregarding special agility applications, the most demanding of the standard slew maneuvers which a spacecraft can perform in LEO is typically ground-tracking, for instance, to point a directional antenna toward a ground station. Considering a worst-case orbital altitude of 300 km, which is the lowest a CubeSat can expect to be operational at for any significant amount of time, the maximum required slew rate is about 1.5 deg/s (for a 90° elevation pass). Using a 2x or 3x margin factor, it is therefore typically acceptable to aim at 3–5 deg/s maximum slew rate on any single axis. The resulting momentum requirement will depend naturally on the spacecraft moments of inertia: a small 1U (if it needs wheels at all) can accept much less than 1 mNm, while 3U and 6U CubeSat are typically satisfied with wheels in the 5 to 10 mNm class; 12U with large deployables might need up to 50 mNm to attain full agility.

Most CubeSat-size COTS reaction wheels indeed cover this range, with products spread along the spectrum from 2 to 100 mNm, at maximum rotor speeds of a few thousand RPMs.

As mentioned, peak torque is typically not the limiting factor in selecting a reaction wheel, and most CubeSat COTS attain 1 to 20 mNm. For most CubeSat wheels, the ratio between momentum capacity and peak torque is in the ballpark of 10 s, which is therefore also the characteristic timescale for a full spin-up from stationary to saturation. Note that, due to the torque characteristic of BLDCs, the actual motor torque will tend to decrease linearly with the wheel speed, so that peak torque is only achieved at low speeds.

Other secondary reaction wheel specs which can be relevant to the system design are the rotor static/dynamic unbalance, and consequently jitter performance. This is especially important for applications requiring high stability, such as long exposure, or narrow FOV imaging.

Finally, of course, one should select a wheel with good power efficiency, acceptable mass and envelope, and, above all, adequate lifetime qualification.

As already mentioned, reaction wheels are prone to various failure modes and shall be considered as one of the critical/limiting items when assessing the design lifetime of a CubeSat. We present here a concise, though not exhaustive, list of the main pitfalls and criticalities which designers need to know about when dealing with these components:

- *Balancing.* As said, balancing is very important when assessing jitter. More in general, though, the balancing of a reaction wheel also determines to an extent the level of stress and wear on its axle and bearings, and therefore its lifetime. Balancing a CubeSat reaction wheel rotor is a very specialized process and requires attaining very tight tolerances, given the size of the parts involved. A wheel with a flatter form factor, thus with a higher inertia rotor, might be more efficient in achieving high momentum storage at moderate wheel speeds and rotor masses, but it will be harder to balance and keep in alignment, given the shorter axle. On the contrary, taller rotors can be supported and balanced more easily, but it will be less mass- and energy-efficient.
- *Zero-crossing.* Commanding a reaction wheel at very low speeds is generally undesirable due to a number of reasons, from increased jitter, stress, and nonlinearities due to stiction, to higher currents given the reduced counter-electromotive force, and even to singularities in certain BLDC controllers exploiting back EMF. For this reason, wheel speed control algorithms should strive to minimize the number of zero-crossing events. It is, for instance, a good practice to set deadbands on the desaturation control, in order not to bring the wheels speed excessively toward zero and force a lot of zero-crossing events to occur.
- *Inductive kickback/recirculation.* Switching the windings of the BLDCs on and off will naturally cause inductive kickbacks, as the current in the windings can't change instantaneously due to their inductance, and the counter-electromotive force can itself induce voltages in the coils. Virtually all BLDC controllers nowadays make use of flyback diodes and resistors to dissipate these currents or to induce braking torques when required, so this issue is typically not of particular concern for the COTS user. Some designs can also support regenerative braking, i.e., converting the flywheel mechanical energy into electrical energy which is pushed back into the supply lines. While this could be in principle beneficial for the power budget, it would need to be supported by the EPS, since not all buses can support generator loads; actually, most battery-based CubeSat EPS are ill-equipped to support such functionality and using regenerative braking could generate undesirably high voltages and noise on the main bus, and potentially even damage other subsystems. One should therefore pay extra care in matching the EPS capabilities with the reaction wheels control circuitry requirements.

Magnetic torquers

More than any other subsystem presented in this section, magnetic torquers are the ones for which even a first-time CubeSat manufacturer might look into the option of producing them in house. While there are, of course, a large number of commercially available magnetorquers from various CubeSat and subsystems manufacturers, built and tested to high quality standards, they are essentially nothing more than simple solenoids, there's really no "secret sauce" to them. Buying a COTS version can surely save time and manpower but will typically not save cost.

There are mainly three kinds of magnetorquers:

- *Vacuum core*. They are typically solenoids with a large (and mostly empty) cross-sectional area and short length—they are the easiest to design and assemble, though they offer a lower performance density (in terms of volume, mass, and power). Among their advantages they offer a simpler control, being their output moment quite linear with the supplied current, and while bulky they can often fit around other components or be run along the edges of the structure. They tend to be used on smaller form factor CubeSats.
- *Printed circuit board (PCB)-embedded torquers*. They are sets of spiral traces embedded into the vehicle PCBs and acting as planar solenoids. They offer the lowest profile and even the possibility to integrate them in the internal layers of solar panel PCBAs. Due to intrinsic limitations given by the number of PCB layers and the trace resistance, they can offer only very limited authority and are therefore usually not seen in designs larger than a 1U.
- *Ferromagnetic core torquers*. They are rod-shaped solenoids wound around a cylindrical core of soft ferromagnetic material, which significantly boosts the generated magnetic moment. The drawback is a slightly more complex control, in order to keep the residual magnetization of the core monitored. Given their highest performance density with respect to the other designs, they have become the de-facto standard for most commercial magnetorquers. COTS are available from 0.1 Am^2 up to even 50 Am^2 and beyond (for micro- and mini-platforms).

Magnetorquers have a single main design parameter, which is the magnetic moment they can generate, measured in Am^2 . As secondary parameters, the designer should take care to match their electrical requirements to the EPS/magnetorquer driver design, since they require a minimum supply voltage to reach the design magnetic moment, and they shall be typically piloted in current.

For detumbling and desaturation tasks, the required control authority for a magnetorquer is constrained on its lower bound by the maximum allowed detumbling time and the maximum expected external disturbance torque.

In the former case, the allowed detumbling time is limited by the available time from separation before the spacecraft needs to achieve a stable attitude—e.g., Sun pointing; for CubeSats, ideally the spacecraft should be able to be power positive even when tumbling, so more often than not the detumbling time is fixed at a more arbitrary/practical threshold, which can be in the order of a few tens of minutes to few hours for higher performance CubeSats, and a few days for lower performance ones (e.g., educational 1Us). A target value in the ballpark of 10^4 seconds is an acceptable starting point, though aiming at something closer to 10^3 seconds, if the power allows it, is preferable. The other figures one needs to calculate the required control authority are the CubeSat moment of inertia and the tumbling rate at deployment. While the former can be easily derived from design information, the latter depends on several factors, such as the deployment method, the CubeSat form factor, the distance between the geometric center and center of mass, the dispenser tolerances, the launch vehicle conditions at deployment, etc. Deployment from a dispenser can typically induce tumble rates of less than 1 deg/s to 10 deg/s, while hand deployment by astronauts and cosmonauts from the ISS have been shown to induce rates even in excess of 200 deg/s [178] — although this is somewhat of an edge case.

Now, the characteristic time of the detumbling phase can be easily shown by dimensional analysis to be proportional to the following:

$$\tau \sim \frac{\|I\omega\|}{\|\mathbf{m} \times \mathbf{B}\|} \sim \frac{\|I\omega\|}{\|\mathbf{m}\| \|\mathbf{B}\|}$$

where the numerator terms are the tensor of inertia and the angular velocity, while the denominator terms are the magnetic moment and the geomagnetic field.

This proportionality is valid up to a constant term including the orbit averaging of the magnetic field value, the variations and cosine losses of the magnetorquer effect, and the efficacy of the detumbling algorithm employed (cfr. [Chapter 10](#) — Control and [Chapter 14](#) — Applicative GNC Cases and Examples). An intuition for this formula is that the numerator is the angular momentum to dump, while the denominator is an estimate of the maximum torque which can be generated at any time; a good detumbling algorithm will achieve an average torque not much smaller

than this—maybe 50% of it when averaging over the orbit, but surely within the same order of magnitude.

Let's assume reference CubeSats moment of inertia figures from [Table 15.6](#) and a reference value of 10 deg/s initial tumble rate. Let's furthermore assume a conservative 200 mG reference magnetic field value (typical low bound in LEO—[\[179\]](#)). With 1 h target detumbling time, we can calculate the ballpark magnetic moment required, reported in [Table 15.7](#):

Moving on to the maximum external disturbance torque, this is the effect which needs to be counteracted by the torquers to allow for reaction wheels desaturation. Estimating the sources of this torque can be tricky and is in general very dependent on the specific CubeSat form factor, architecture, and design. Short of a detailed torque budget, however, some ballpark estimates can still be achieved, assuming worst-case figures. The main contributions are:

- Magnetic, from the residual dipole of the spacecraft, and its varying components due to power lines, harness, and solar arrays; this is effectively a “noise” which needs to be added to the magnetorquers effect. Magnetic cleanliness practices can be employed to minimize this noise, including degaussing, using twisted pairs in power harnessing, avoiding current loops in the boards layout, avoiding the use of ferromagnetic materials, etc. [\[180\]](#). Still, a residual dipole in the ballpark of 10^{-2} A m² (inducing up to 10^{-6} Nm) would not be unusual. In this sense, for instance, sizing the magnetorquers for a 1U just based on the detumbling requirement presented previously could result in undersized torquers and an excessive residual dipole versus magnetorquer moment ratio.
- Aerodynamic, from asymmetrical effects of thermospheric drag in LEO. This is highly dependent not only on the CubeSat geometry but also on its orbital altitude and current solar cycle activity. It can be as low as 10^{-10} Nm for small symmetrical CubeSat at high altitudes in periods of solar minimum, as well as up to 10^{-5} Nm for large CubeSats with extensive deployable appendages at ISS altitudes during solar maximum.

Table 15.7 Estimate of magnetic moment for CubeSat detumbling.

Form factor **Required magnetic moment for detumbling**

1U	0.005 A m^2
3U	0.10 A m^2
6U XL	0.45 A m^2
12U XL	0.94 A m^2

Table 15.8 Recommended magnetorquer for standard CubeSats.

Form factor	Recommended magnetorquer
1U	0.10 A m^2 (2 uNm)
3U	0.10 A m^2 (2 uNm)
6U XL	0.50 A m^2 (10 uNm)
12U XL	1.00 A m^2 (20 uNm)

- Radiative, given by the effects of absorbed and reflected incoming radiation, as well as emitted thermal radiation, and dependent on the space-craft geometry and variation of surface optical coefficients. This is typically significant only for CubeSats with deployable arrays, whenever large asymmetrical sections of these are shadowed and would still be well below 10^{-6} Nm.
- Gravitational, given by the effect of the gravity gradient. Except for CubeSats with very large deployable solar arrays, this contribution will always be well below 10^{-6} Nm, with more typical values around 10^{-7} Nm.

The available magnetic moment should be at least one order of magnitude greater than the RMS of the disturbance torques, assuming worst-case magnetic field intensity (200 mG in LEO) [179]:

$$\mathbf{m} > 10 \frac{\|\mathbf{t}_{dist}\|}{\|\mathbf{B}_{min}\|}$$

In general, for smaller vehicles, the uncertainty on the residual dipole will be the limiting factor for determining the minimum magnetorquer authority, while for larger vehicles, the detumbling torque is already more than enough to also overcome any disturbance torque (Table 15.8).

On the subject of piloting the torquers, a simple bang–bang control with a central deadband usually grants sufficient performance for most applications, i.e., detumbling and wheel desaturation, which have loose actuation accuracy requirements and slow dynamics (even in the order of 10^2 or 10^3 seconds), see Chapter 10 — Control. The designer shall take into account that using current modulation schemes can introduce switching losses (for PWM or pulse-density modulation) or regulation losses (for linear current regulation from a constant voltage source), which in naive implementations could even offset the increased efficiency provided by more sophisticated control laws. Unless magnetic actuation is used as the primary attitude control source, or in other special cases in which more accurate

magnetic pointing/control requirements are present, the improvement in performance introduced by modulating the current in the rods may not be worth the added complexity; simplicity and robustness are typically preferred features when dealing with magnetorquers. This is not to say that more sophisticated control laws can't be implemented effectively, but rather that satisfactory results can already be achieved with the simplest approach.

On the other hand, it is important to foresee at least some amount of periodic switching of the magnetorquers, in particular, to avoid disturbing the already noisy magnetometers measurements. It is, therefore, a common practice to periodically alternate between a magnetorquer actuation window, in which the required magnetorquers can be powered, and then a magnetometer measurement window, in which all magnetorquers are depowered and the magnetometers are sampled. The selection of the period and duration of these windows needs to maximize magnetorquers duty cycle, while also allowing for enough time to perform and average the required measurements, and at a sufficient rate to capture the highest frequency magnetic field variation which one expects to see (driven chiefly by the maximum expected tumble rate).

GNC system example

This section aims at giving to the reader an example of a possible CubeSats GNC architecture stemming from the suggestions provided above in this chapter. In particular, it will focus on the specific case of the ADCS of a LEO CubeSat. Starting from a list of high-level specifications, the overall architecture of the GNC subsystem will be presented, providing some details on the accommodation and on the various figures related to the different sensors and actuators selected for the spacecraft.

It is important to mention that this section will not provide details on the verification of the high-level requirements used to derive the architecture. However, as thoroughly discussed in [Chapter 12](#) — GNC Verification and Validation, the aspiring CubeSat manufacturer shall always verify each requirement with the correct method and so ensure the suitability of the selected design with the mission.

[Table 15.9](#) reports the specifications/requirements of the example CubeSat applicable to the ADCS.

Considering all the various requirements applicable to the ADCS, a preliminary architecture of the CubeSat ADCS can be derived.

Table 15.9 Specifications and requirements of the example CubeSat applicable to the ADCS.

ID	Requirement
EXP-REQ-1	The CubeSat shall be compliant with the standard 3U form factor
EXP-REQ-2	The payload boresight shall be directed as ZP axis on the CubeSat body-fixed reference frame (see Fig. 15.56)
EXP-REQ-3	The solar panels main direction shall be directed as YP axis on the CubeSat body-fixed reference frame (see Fig. 15.56)
EXP-REQ-4	The S-band antenna boresight shall be directed as YM axis on the CubeSat body-fixed reference frame (see Fig. 15.56)
EXP-REQ-5	The CubeSat shall be able to autonomously perform detumbling in less than 1 h starting from an angular rate initial condition of 5 deg/s
EXP-REQ-6	While in Sun pointing, the ADCS shall be able to maintain the absolute pointing error of the orthogonal vector of the solar array with respect to the Sun-S/C vector lower than 5 deg with 99.7% probability at 90% confidence level
EXP-REQ-7	During payload operations, the ADCS shall be able to maintain the absolute pointing error of the payload boresight vector with respect to the nadir vector lower than 1 deg with 99.7% probability at 90% confidence level
EXP-REQ-8	During payload operations, the ADCS shall be able to guarantee a relative pointing error of the payload boresight vector below 0.1 deg over 100 ms time window with 99.7% probability at 90% confidence level
EXP-REQ-9	During communication windows, the ADCS shall be able to maintain the absolute pointing error of the S-band antenna boresight vector with respect to the ground station-S/C vector lower than 2 deg with 99.7% probability at 90% confidence level
EXP-REQ-10	The CubeSat shall be designed to operate in a Sun-synchronous orbit with altitude of 550 km and LTDN of 10:30

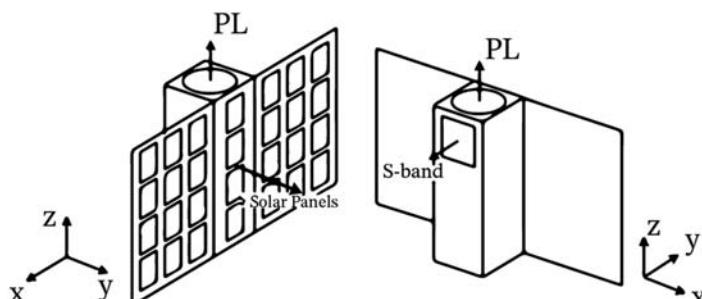


Figure 15.56 Example CubeSat architecture constraints.

Here it follows a list of useful components on-board the spacecraft in order to satisfy the various requirements:

- An IMU is needed on-board the CubeSat to measure the body rates of the vehicle at the deploy as well as during nominal mission.
- At least three reaction wheels are needed to perform three-axis stabilized attitude control during nominal mission and spacecraft detumbling.
- A set of magnetometers can be exploited to obtain a rough estimate of the attitude of the vehicle. This could serve as the main source of attitude determination during degraded states (contingency modes) or can be used in conjunction with more precise sensors during nominal mission.
- At least three magnetorquers are needed to perform RWs desaturation. Moreover, they could be used to perform attitude control during degraded states (contingency modes).
- A set of Sun sensors are useful to obtain a rough estimate of the relative position of the Sun in the body frame. This could serve as the main attitude determination sensor during degraded states (contingency modes) or can be used in conjunction with more precise sensors during nominal mission.
- More precise attitude sensors such as star trackers or horizon sensors are needed in order to fulfill the pointing requirements of the mission.
- A GNSS receiver module needs to be mounted on the spacecraft to have inertial position and velocity knowledge, useful in all the pointing modes related with PL and S-band operations.
- Finally, a dedicated PCBA able to interface with all the various sensors and actuators is needed, together with a processor running the ADCS algorithms. This could be the same processor used for the on-board data handling or, in some cases, a dedicated piece of HW for specific and demanding applications.

The technology of the sensors/actuators, the number of modules and their accommodation need to be optimized considering various factors:

- The mass budget of the vehicle.
- The power budget of the spacecraft.
- The various pointing constraints and accommodation constraint of the different components versus the available surface of the spacecraft.
- The thermal analysis and the operative/nonoperative temperature ranges of the components.
- The operative ranges of the different modules with respect to the predicted mission scenario.
- The available budget of the mission.

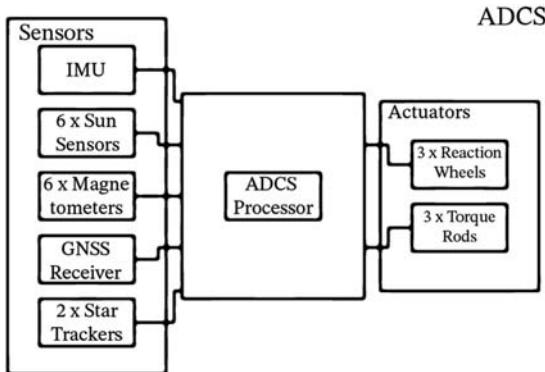


Figure 15.57 CubeSat ADCS architecture example.

The diagram in Fig. 15.57 reports a possible architecture for the ADCS of the LEO 3U CubeSat under consideration. It is important to mention that this is not the only one possible architecture, and the reader should only consider it as a guideline. However, due to the standardization of the CubeSat platform and the level of maturity of technology exploited, the solution proposed will not deviate much from the possible alternatives.

Inertial measurement unit

The selected IMU is based on three-axes gyro MEMS, for a very miniaturized module. It is usually mounted in the central part of the vehicle, paying attention to its alignment with the body axes. Below are some reference figures:

- Range: $\pm 400^\circ/\text{s}$.
- Bias instability: $0.3^\circ/\text{h}$.
- Angular random walk: $0.15^\circ/\sqrt{\text{h}}$.
- Bias error over temperature: $9^\circ/\text{h}$.

Sun sensors

Being a secondary sensor used for contingency scenarios, full coverage is preferred with respect to resolution. In fact, six photodiode-based Sun sensors are considered for the CubeSat. They are very small in size so to be placed on each external face of the vehicle. In this way, it is always possible to completely reconstruct the information of the relative position of the Sun with respect to the body frame. It is important to pay attention to the orientation of the sensor with respect to the body frame as well as to not mount them in shadowed areas.

Hereunder, the main figures related to the selected Sun sensors:

- Range: $\pm 75^\circ$.
- Resolution: 2.7° .

Magnetometers

The LEO CubeSat under consideration is also equipped with six three-axes magnetometers needed for the reconstruction of the Earth magnetic field, useful for desaturation algorithms as well as for very simple control algorithms (e.g., B-dot, see [Chapter 10](#) – Control and [Chapter 14](#) – Applicative GNC Cases and Examples).

It is a good practice to place at least one sensor per each vehicle face, mounted externally, with their reference frame aligned with the body frame and as far as possible from the main perturbing components (battery module, torque rods, and reaction wheels).

Sensor fusion and automatic calibration using Earth magnetic models are suggested to the reader in order to obtain a more reliable and robust measurement, see [Chapter 14](#) – Applicative GNC Cases and Examples.

The main specifications of the selected magnetometers are:

- Range: ± 1 G.
- Resolution: 0.73 mG.

Star trackers

In order to fulfill the pointing requirements during PL operations and S-band sessions, a set of star trackers is exploited. A good practice is to use two star trackers, orthogonally mounted one to each other (so to reduce the around-boresight error) and facing away from the Sun during Sun-pointing modes as well as facing away from the Earth limb during PL operations. Typical figures related to star trackers for small CubeSats are:

- FOV: $10^\circ \times 20^\circ$.
- Accuracy cross-boresight: 0.01° (3-sigma).
- Accuracy around-boresight: 0.05° (3-sigma).
- Sky coverage: >95%.
- Occlusion angle with the Sun: $\pm 70^\circ$ (from boresight).
- Occlusion angle with the Earth limb: $\pm 35^\circ$ (from boresight).

Thus, a possible configuration for the CubeSat would be to mount one star tracker on the ZM face with the boresight directed as -Z axis and the other star tracker mounted on the XM face with the boresight laying on the XZ plane, tilted 15° away from -X toward -Z (see [Fig. 15.58](#)).

In this way, at least one star tracker would be always able to face the dark space during both Sun-pointing modes and PL operations.

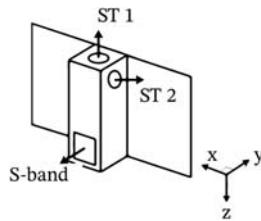


Figure 15.58 Star trackers orientation on example CubeSat.

GNSS receiver

In a LEO CubeSat, the inertial knowledge is very beneficial, especially during ground target modes or nadir-pointing modes when the spacecraft is equipped with star trackers instead of Earth sensors. Therefore, a GNSS receiver is integrated in the vehicle to get information of position and velocity in ECEF coordinates.

Common figures for a GNSS receiver are listed below:

- Signal tracking: GPS L1.
- Position accuracy: 1.5 m RMS.
- Velocity accuracy: 0.03 m/s RMS.
- Time accuracy: 20 ns RMS.

Particular attention shall be placed on the orientation of the GPS antenna which shall always face away from the Earth in order to optimize its visibility toward the GPS constellation. Thus, in the CubeSat under consideration, it can be mounted on the XM face with its boresight aligned with the -X axis.

Reaction wheels

In order to actively control all the three axes of the spacecraft, a set of three orthogonally mounted reaction wheels is employed in the vehicle.

No redundancy is selected in this case for few reasons:

- Mass and volume limitations of a 3U CubeSat usually do not allow for redundant modules.
- Power limitations of a 3U CubeSat typically limits the possibility of having hot redundancies.
- Typical lifetime for CubeSat missions is in the order of months, and the associated risk is usually higher with respect to other kind of missions: the risk of having only three wheels can usually be accepted.

The reaction wheels are mounted on the central part of the vehicle, paying attention to align them with the body axes.

They need to be properly selected, depending on the mission scenario, the available power, mass and volume, the required robustness, and especially the required stability. Below are few specifications of possible wheels for a 3U CubeSat:

- Speed range: $\pm 13,000$ rpm.
- Maximum torque: 2.5 mNm.
- Maximum momentum: 15 mNms.

Magnetorquers

Reaction wheels alone are not enough and a secondary set of actuators, able to dump away the stored momentum, is needed. In LEO CubeSat, it is very popular to have coils or magnetic torquers which are capable of creating a torque able to desaturate the flying wheels. They are very simple actuators and usually quite reliable; therefore, it is enough to have three of them, orthogonally mounted and possibly aligned with the reaction wheel axes.

As shown above in this chapter, a typical figure for a 3U CubeSat torque rod is:

- Magnetic momentum: 0.1 A m^2 .

Verification and testing limitations

The following section gives a tailored nanosatellite perspective on the content thoroughly described in [Chapter 10](#) – Control and [Chapter 12](#) – GNC Verification and Validation.

V&V processes have always been a fundamental basis for all the space missions up to the advent of the CubeSats. In fact, if on one hand, the nanosatellites are revolutionizing the space industry, on the other hand, they are also moving it toward a riskier direction. When speaking about CubeSats, mission failure is actually an option: this is mainly due to the fact that the actors involved in this type of missions (universities and small companies) are usually more focused on the development rather than the verification and testing. As a consequence, considering more than 250 CubeSats launched in the last years, about 50% didn't succeed in their mission objectives [181]. Over time, the platforms are gaining flight heritage and so the companies which are specialized in this space domain. As a result, the infant mortality of the launched CubeSats passed from 100% in 2002 to 23.3% in 2018 [182]. This is also due to the consolidation of processes and procedures able to lower the risk of mission failure in the CubeSat industry.

The first important element needed during the definition of a space mission which is also a good starting point for V&V is a comprehensive

set of mission requirements and system requirements, together with mission objectives and goals.

Specifically, for the ADCS, several references are available in literature which can be used to derive engineering requirements (e.g., Tailored ECSS Engineering Standards for In-Orbit Demonstration CubeSat Projects [152]). Pointing requirements are among the most discussed because there is a high degree of unrealistic expectations. Pointing performances of large science missions are not representative of generic bus performances, and they would require ad hoc sensors and solutions to be achieved. Moreover, simple specifications in terms of allowed pointing errors are not enough and a well-written requirement shall be able to completely specify all the scenarios under regulation. This is also a benefit during V&V, serving as guideline for the testing procedures.

A satisfactory pointing requirement would at least contain, see [Chapter 10](#) – Control and [Chapter 12](#) – GNC Verification and Validation:

- Indications on the type of error being regulated, such as absolute pointing error, relative pointing error, absolute knowledge error, etc. [153].
- The mission phase during which it is expected to be satisfied (e.g., during PL operations, during Sun pointing).
- The reference frame in which the error is expressed.
- The allowed pointing error and, in case of relative errors, the related time window.
- The probability level associated to the pointing error.
- The confidence level associated to the pointing error and its probability.

Moreover, it is suggested to express the confidence level in percentile terms rather than sigma terms, due to the non-Gaussian behavior of the functions described.

Last but not least, each requirement shall be characterized by one or more verification methods, intended to be used during the project to validate the system. The most important verification methods available are listed below [183]:

- *Analysis.* Theoretical or empirical evaluations are used to validate the requirement in the target scenario. It falls under the analysis category also the verification by similarity, which exploits the flight heritage of the component to demonstrate its applicability to the mission.
- *Review of design.* Approved design documents and technical material can be used to verify requirements when they are able to unambiguously show that the requirement is met.

- *Inspection*. It can be used for the verification of physical properties or other aspects of the mission (e.g., SW) which can be reviewed without the need of additional equipment or testing HW.
- *Test*. It is the verification method used when measurements of performances and functional properties of the system or its components are possible in a simulated environment. It usually requires ground support equipment when performed in laboratories. Demonstration in orbit also falls under the category of verification by testing.

It is very easy to understand that the most representative and explanatory verification method is by *test*, which shall always be preferred when possible. However, testing the satellite HW and SW in a representative environment can be very challenging and costly.

This is also applicable to the ADCS of CubeSat: verification of ADCS performances on nanosatellites is mainly performed by analysis and demonstration for recurrent platforms. In fact, testing the complete subsystem on-ground with techniques such as hardware-in-the-loop is particularly expensive, complex, and time-demanding. Most of the time it's just not worth it, especially without in-orbit data to validate it. However, there are several other means to be used in order to mitigate the risk of failure in-orbit. Hereafter, few of them are presented and briefly described.

Software-in-the-loop

The ADCS SW can be validated before launch, thanks to dedicated software-in-the-loop testing campaigns. Basically, the flight code is wrapped into a truth model able to recreate all the interfaces of the GNC SW. It simulates the S/C dynamics, the environment, and the behavior of all the various sensors and actuators. Then, thanks to MC simulations, the flight SW is validated against nominal and not nominal conditions to spot any possible bug or nonconformance. Usually, different mission scenarios are simulated. Among them:

- All the contingency modes of the ADCS to ensure its capability to isolate any failure and stabilize in a reduced configuration.
- The Sun-pointing modes, so to verify the correct behavior of the platform in pointing the solar panels to the Sun in a reliable manner.
- The various payload-pointing modes, considering also ground target-pointing modes for the communication system during the downlink sessions.

The results of all these simulations are also very beneficial during the verification of the mission and system requirements, being the result of dedicated analyses on the ADCS.

Hardware performance tests

As mentioned before, overall ADCS tests used to characterize the behavior and the associated performances of the entire system are quite challenging and usually they are not affordable from the small CubeSat manufacturers. Nevertheless, being able to measure the associated characteristics of the various components of the ADCS system is definitely mandatory to ensure their compatibility with the mission.

For this reason, a series of tests can be performed on the HW:

- *Star trackers.* The performances of a star camera can be characterized before launch, thanks to a dedicated campaign on-ground, where the sensor is exposed for a prolonged time to the night sky of a rural area. In this way, the sensor, the associated algorithm, and the star catalog are tested measuring the number of matches and the related accuracy. A good practice is to have two sensors placed in a known configuration, using the first one as the source of truth while the second being under characterization. Another important aspect to consider during ground testing and verification of optical instruments is related to the mechanical misalignments with respect to the body reference frame. Preflight calibration and alignment procedures (e.g., geometric laser alignment) can be adopted; however, they are usually not performed during CubeSats assembly and integration. In fact, any calibration on-ground — while valuable — would be invalidated by launch vibrations. It is surely important for on-ground calibrations to run pre- and postvibe alignment tests, to assess the impact of the environmental campaign. On the other hand, it's paramount to prepare a plan for in-orbit calibration. These considerations are also valid for optical payloads, which tends to impose severe pointing requirements to the platform.
- *Magnetometers.* Dedicated calibration campaigns are needed for this kind of sensors, which are usually cheap and affected by biases and misalignments with respect to the declared reference frame. A possible performance test can consist of the acquisition of a sufficient number of readings from the magnetometer, in various system configurations (to ensure no EMC issues) and then the comparison with an external reliable and accurate probe. In this way, the biases associated with the sensor can be estimated and considered in the postprocessing SW.

- *Sun sensors.* Also, the Sun sensors are usually small components affected by biases and misalignments. Therefore, dedicated calibration campaigns are beneficial, with the aim to characterize the bias error in the sensor measurement. Furthermore, one of the main issues with the Sun sensors is related with the detection of false measurements related to the Earth albedo or other light sources. Thus, if the sensor provides info on the intensity level of the incident light, multiple tests can be performed under the sunlight in order to characterize the minimum and the maximum threshold for the detection of true readings (please consider the atmosphere absorption and scattering in the solar irradiance at the Earth surface with respect to the level the satellite will actually foresee in orbit).
- *Reaction wheels.* These actuators are usually prone to failure in-orbit due to their nature. In fact, mechanisms in space are always analyzed with particular care because they tend to fail with prolonged use. Accelerated life-time testing is crucial in the characterization of the failure modes related to these actuators in-orbit. Examples of possible tests are: thermal cycling, vibrations, wheel speed cycling, and zero-crossing. Another important aspect related to the reaction wheels performances is the static and dynamic unbalance of the flying wheel which could impact on the spacecraft relative pointing error. A six-axis Kistler table can be exploited on-ground during a dedicated testing campaign to characterize the wheel behavior [184].

Hardware functional tests

If the HW performance tests are able to characterize the various components of the ADCS, they are usually not able to verify the overall system and to assess any incompatibility between the different modules. For this reason, the CubeSat manufacturer shall always consider dedicated HW functional tests in the verification plan. Hereunder, some possible functional tests to be performed before launch:

- *Polarity tests.* It is of paramount importance in the ADCS system to take care about the polarity of the sensors and actuators. In fact, it can be very easy to correctly align the component reference frame with the body-fixed reference frame without respecting the correct directions. For example, the torque rods could be aligned with the body reference frame, but their polarity could be erroneous simply due to swapped electrical wires. Thus, dedicated polarity checks can be exploited to highlight assembly and/or configuration issues before launch.

- *Day-in-the-life test.* It is a very good practice to run a prolonged test of the overall ADCS and system SW, running on the target HW, to spot possible bugs in the flight SW and related configurations. In fact, even if the sensors and the actuators are not subjected to the actual scenario that they will encounter during the mission, it is usually possible to test at this stage the various logics and mechanisms related to the GNC state machines, the FDIRs, and other parts more related to sensors post-processing and health checks.

Hardware-in-the-loop

Sometimes, SW testing and components testing is not enough, for example, due to the very complicated nature of the mission (e.g., formation flying, rendezvous, and docking). In these cases, very particular and dedicated ground equipment is needed. Off-loading systems, able to simulate a force-less and torque-less environment, exist in many universities and research centers; however, they are usually limited in terms of HW that can be host, dimensions, mass, and volume.

They are usually based on air bearings, mechanical devices able to create a thin film of air between two surfaces in relative motion, thus reducing the friction between moving parts by several orders of magnitude. Depending on the degrees of freedom available in the simulator, they can be divided in mainly three categories [185]:

- Planar systems, used to recreate a force-less environment which can be useful in simulating the orbital dynamics during rendezvous and docking.
- Rotational systems for the test and validation of the ADCS.
- Combination systems able to provide to the payload 5 DOFs: all the possible movements of a rigid body except for the translations normal to the testbed floor. In this way, they recreate the environment needed during formation flying experiments.

References

- [1] A.L. Samuel, Some studies in machine learning using the game of checkers, IBM Journal of Research and Development 3 (3) (1959) 210–229, <https://doi.org/10.1147/rd.33.0210>.
- [2] T.M. Mitchell, Machine Learning, 1997, <https://doi.org/10.1109/ICDAR.2019.00014>.
- [3] I.T. Jolliffe, Graphical representation of data using principal components, Principal Component Analysis (2002) 78–110.
- [4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, D. Hassabis, Mastering the game of Go without human knowledge, Nature 550 (7676) (2017) 354–359, <https://doi.org/10.1038/nature24270>.

- [5] J. MacQueen, Some methods for classification and analysis of multivariate observations, *Berkeley Symposium on Mathematical Statistics and Probability* 1 (14) (1967) 281–297, <https://doi.org/10.1007/s11665-016-2173-6>.
- [6] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 2000, ISBN 978-1-4757-3264-1.
- [7] D.E. Rumelhart, G.E. Hinton, Learning representations by back-propagating errors, *Cognitive Modeling* 2 (2019) 3–6, <https://doi.org/10.7551/mitpress/1888.003.0013>.
- [8] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Networks* 5 (2) (1994) 157–166, <https://doi.org/10.1109/72.279181>.
- [9] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, *30th International Conference on Machine Learning, ICML 2013 (PART 3)* (2013) 2347–2355.
- [10] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, *Proceedings—IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2016) 770–778, <https://doi.org/10.1109/CVPR.2016.90>.
- [11] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural Computation* 9 (8) (1997) 1735–1780, <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [12] S. Silvestrini, M. Lavagna, Neural-aided GNC reconfiguration algorithm for distributed space system: development and PIL test, *Advances in Space Research* 67 (5) (2021) 1490–1505, <https://doi.org/10.1016/j.asr.2020.12.014>.
- [13] S. Silvestrini, M. Lavagna, Neural-based predictive control for safe autonomous spacecraft relative maneuvers, *Journal of Guidance, Control, and Dynamics* 44 (2021) 2303–2310, <https://doi.org/10.2514/1.G005481>.
- [14] V. Pesce, S. Silvestrini, M. Lavagna, Radial basis function neural network aided adaptive extended Kalman filter for spacecraft relative navigation, *Aerospace Science and Technology* 96 (2020) 105527, <https://doi.org/10.1016/j.ast.2019.105527>.
- [15] M. Bechini, et al., Tango Spacecraft Dataset for Region of Interest Estimation and Semantic Segmentation, 2022, <https://doi.org/10.5281/zenodo.6507863>, 1–1.
- [16] H. Lee, R. Grosse, R. Ranganath, A.Y. Ng, Unsupervised learning of hierarchical representations with convolutional deep belief networks, *Communications of the ACM* 54 (10) (2011) 95–103, <https://doi.org/10.1145/2001269.2001295>.
- [17] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016.
- [18] M.Z. Asghar, M. Abbas, K. Zeeshan, P. Kotilainen, T. Hämäläinen, Assessment of deep learning methodology for self-organizing 5G networks, *Applied Sciences* 9 (15) (2019) 2975, <https://doi.org/10.3390/app9152975>.
- [19] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, *Communications of the ACM* 60 (6) (2012) 84–90, <https://doi.org/10.1145/3065386>.
- [20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, et al., Imagenet large scale visual recognition challenge, *International Journal of Computer Vision* 115 (3) (2015) 211–252.
- [21] E. Shelhamer, J. Long, T. Darrell, Fully convolutional networks for semantic segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (4) (2017) 640–651, <https://doi.org/10.1109/TPAMI.2016.2572683>.
- [22] O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, *Medical Image Computing and Computer-Assisted Intervention—MICCAI* (2015) 234–241, https://doi.org/10.1007/978-3-319-24574-4_28.

- [23] Y. Cai, L. Qian, Y. Fan, L. Zhang, H. Huang, X. Ding, An automatic trough line identification method based on improved UNet, *Atmospheric Research* 264 (2021) 105839, <https://doi.org/10.1016/j.atmosres.2021.105839>. ISSN 0169-8095.
- [24] Z.Q. Zhao, P. Zheng, S.T. Xu, X. Wu, Object detection with deep learning: a review, *IEEE Transactions on Neural Networks and Learning Systems* 30 (11) (2019) 3212–3232, <https://doi.org/10.1109/TNNLS.2018.2876865>.
- [25] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587, <https://doi.org/10.1109/CVPR.2014.81>.
- [26] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (6) (2017) 1137–1149, <https://doi.org/10.1109/TPAMI.2016.2577031>.
- [27] K. He, G. Gkioxari, P. Dollar, R. Girshick, Mask R-CNN, *Proceedings of the IEEE International Conference on Computer Vision* (2017) 2980–2988, <https://doi.org/10.1109/ICCV.2017.322>.
- [28] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788, <https://doi.org/10.1109/CVPR.2016.91>.
- [29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, A.C. Berg, SSD: single shot multibox detector, *Lecture Notes in Computer Science* (2016) 21–37, https://doi.org/10.1007/978-3-319-46448-0_2, 9905 LNCS.
- [30] T.Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42 (2) (2020) 318–327. <https://doi.org/10.1109/TPAMI.2018.2858826>.
- [31] Y. Yin, H. Li, W. Fu, Faster-YOLO: an accurate and faster object detection method, *Digital Signal Processing* 102 (2020) 102756, <https://doi.org/10.1016/j.dsp.2020.102756>.
- [32] H. Maudi Lathifah, et al., Fast and accurate fish classification from underwater video using you only look once, *IOP Conference Series: Materials Science and Engineering* 982 (2020) 012003.
- [33] M. Mozer, A focused backpropagation algorithm for temporal pattern recognition, *Complex Systems* 3 (1989) 349–381.
- [34] A.J. Robinson, F. Fallside, The utility driven dynamic error propagation network, *IEEE Conference (Neural Information Processing Systems)* (1987).
- [35] P.J. Werbos, Generalization of backpropagation with application to a recurrent gas market model, *Neural Networks* 1 (4) (1988) 339–356, [https://doi.org/10.1016/0893-6080\(88\)90007-X](https://doi.org/10.1016/0893-6080(88)90007-X).
- [36] J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, in: *Proceedings of the National Academy of Sciences* 81, 1984, pp. 3088–3092, 10.
- [37] Abe, Theories on the Hopfield neural networks, *International 1989 Joint Conference on Neural Networks* 1 (1989) 557–564.
- [38] M. Atencia, G. Joya, F. Sandoval, Parametric Identification of Robotic Systems with Stable Time-Varying Hopfield Networks *Neural Computing and Applications*, vol 13, 2004, pp. 270–280.
- [39] Y. Hernández-Solano, M. Atencia, G. Joya, F. Sandoval, A discrete gradient method to enhance the numerical behaviour of Hopfield networks, *Neurocomputing* 164 (C) (2015) 45–55.

- [40] A. Pasquale, S. Silvestrini, A. Capannolo, P. Lunghi, M. Lavagna, Small bodies non-uniform gravity field on-board learning through Hopfield Neural Networks, *Planetary and Space Science* (2022) 105425.
- [41] European Cooperation for Space Standardization, ECSS-E-ST-40C—Software, ECSS secretariat, ESA-ESTEC, The Netherlands
- [42] European Cooperation for Space Standardization, ECSS-Q-ST-80C—Software Product Assurance, ECSS Secretariat, ESA-ESTEC, The Netherlands
- [43] European Cooperation for Space Standardization, ECSS-E-ST-60-10C—Control Performance, ECSS Secretariat, ESA-ESTEC, The Netherlands
- [44] European Cooperation for Space Standardization, ECSS-E-ST-60-02C—ASIC and FPGA Development, ECSS secretariat, ESA-ESTEC, The Netherlands
- [45] A. Tatsch, N. Fitz-Coy, S. Gladun, On-orbit servicing: a brief survey, in: *Proceedings of the 2006 Performance Metrics for Intelligent Systems Workshop*, 2006, pp. 21–23.
- [46] M. Wieser, H. Richard, G. Hausmann, J.-C. Meyer, S. Jaekel, M. Lavagna, R. Biesbroek, e.Deorbit Mission: OHB Debris Removal Concepts. *ASTRA 2015-13th Symposium on Advanced Space Technologies in Robotics and Automation*, Noordwijk, 2015.
- [47] S. Sharma, J. Ventura, S. D’Amico, Robust model-based monocular pose initialization for noncooperative spacecraft rendezvous, *Journal of Spacecraft and Rockets* 55 (2018) 1–16, <https://doi.org/10.2514/1.A34124>.
- [48] L. Pasqualetto Cassinis, R. Fonod, E. Gill, Review of the robustness and applicability of monocular pose estimation systems for relative navigation with an uncooperative spacecraft, *Progress in Aerospace Sciences* 110 (2019), <https://doi.org/10.1016/j.paerosci.2019.05.008>.
- [49] S. D’Amico, M. Benn, J. Jorgensen, Pose estimation of an uncooperative spacecraft from actual space imagery, *International Journal of Space Science and Engineering* 2 (2014) 171–189, <https://doi.org/10.1504/IJSPACSE.2014.060600>.
- [50] M. Bechini, et al., Tango Spacecraft Wireframe Dataset Model for Line Segments Detection, 2022.
- [51] M. Kisantai, S. Sharma, T.H. Park, D. Izzo, M. Martens, S. D’Amico, Satellite pose estimation challenge: dataset, competition design and results, *IEEE Transactions on Aerospace and Electronic Systems* (2020), <https://doi.org/10.1109/TAES.2020.2989063>.
- [52] D. Rondao, N. Aouf, Multi-view monocular pose estimation for spacecraft relative navigation. 2018 AIAA guidance, navigation, and control conference, Kissimmee (2018), <https://doi.org/10.2514/6.2018-2100>.
- [53] V. Capuano, S.R. Alimo, A.Q. Ho, S. Chung, Robust Features Extraction for On-Board Monocular-Based Spacecraft Pose Acquisition, *AIAA Scitech 2019 Forum*, San Diego, CA, USA, 2019, <https://doi.org/10.2514/6.2019-2005>.
- [54] J. Dai, Y. Li, K. He, J. Sun, R-FCN: object detection via region-based fully convolutional networks, *Advances in Neural Information Processing Systems* (2016) 379–387.
- [55] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, et al., Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017. ArXiv Preprint. doi:arXiv:1704.04861.
- [56] A. Newell, K. Yang, J. Deng, Stacked Hourglass networks for human pose estimation, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), *Computer Vision—ECCV 2016*, vol 9912, Springer, Cham, 2016, pp. 483–499.
- [57] K. Sun, B. Xiao, D. Liu, J. Wang, Deep high-resolution representation learning for human pose estimation, in: *2019 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, Long Beach, CA, USA, 2019.

- [58] H. Su, C. Qi, Y. Li, L. Guidas, Render for CNN: viewpoint estimation in images using CNNs, Proceedings of the IEEE International Conference on Computer Vision (2015) 2686–2694.
- [59] S. Sharma, C. Beierle, S. D’Amico, Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks, in: IEEE Aerospace Conference, IEEE, Big Sky, MT, USA, 2018, <https://doi.org/10.1109/AERO.2018.8396425>.
- [60] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, 2014 arXiv preprint arXiv:1409.1556.
- [61] S. Mahendra, H. Ali, R. Vidal, 3D pose regression using convolutional neural networks, Proceedings of the IEEE International Conference on Computer Vision (2017) 2174–2182.
- [62] A. Kendall, M. Grimes, R. Cipolla, Posenet: a convolutional network for real-time 6-DOF camera relocalization, Proceedings of the IEEE International Conference on Computer Vision (2015) 2938–2946.
- [63] S. Sharma, S. D’Amico, Pose estimation for non-cooperative spacecraft rendezvous using neural networks, in: 29th AAS/AIAA Space Flight Mechanics Meeting, IEEE, Ka’anapali, HI, USA, 2019, <https://doi.org/10.1109/AERO.2018.8396425>.
- [64] J.F. Shi, S. Ulrich, S. Ruel, Cubesat simulation and detection using monocular camera images and convolutional neural networks, 2018 AIAA Guidance, Navigation, and Control Conference (2018), <https://doi.org/10.2514/6.2018-1604>. Kissimmee.
- [65] S. Sonawani, R. Alimo, R. Detry, D. Jeong, A. Hess, H. Ben Amor, Assistive relative pose estimation for on-orbit assembly using convolutional neural networks, AIAA Scitech 2020 Forum (2020), <https://doi.org/10.1109/AERO.2018.8396425>. Orlando.
- [66] S. Sharma, S. D’Amico, Reduced-Dynamics Pose Estimation for Non-cooperative Spacecraft Rendezvous Using Monocular Vision. 38th AAS Guidance and Control Conference, 2017. Breckenridge.
- [67] F.L. Markley, Attitude error representations for Kalman filtering, Journal of Guidance, Control, and Dynamics 26 (2003) 311–317, <https://doi.org/10.2514/2.5048>.
- [68] K. Black, S. Shankar, D. Fonseka, J. Deutsch, A. Dhir, M. Akella, Real-time, flight-ready, non-cooperative spacecraft pose estimation using monocular imagery, in: 31st AAS/AIAA Space Flight Mechanics Meeting, 2021.
- [69] L. Pasqualetto Cassinis, R. Fonod, E. Gill, I. Ahrns, J. Gil-Fernandez, Evaluation of tightly- and loosely-coupled approaches in CNN-based pose estimation systems for uncooperative spacecraft, Acta Astronautica 182 (2021) 189–202, <https://doi.org/10.1016/j.actaastro.2021.01.035>.
- [70] C. Wiedermann, S. Flegel, M. Mockel, J. Gelhaus, V. Braun, C. Kebschull, P. Vorsmann, Cost estimation of active debris removal, in: 63rd International Astronautical Congress, 2012. Naples, Italy.
- [71] H. Schaub, L. Jasper, P. Anderson, D. McKnight, Cost and risk assessment for space-craft operation decisions caused by the space debris environment, Acta Astronautica (2015) 66–79.
- [72] L. Pasqualetto Cassinis, A. Menicucci, E. Gill, I. Ahrns, M. Sanchez-Gestido, On-ground validation of a CNN-based monocular pose estimation system for uncooperative spacecraft: bridging domain shift in rendezvous scenarios, Acta Astronautica 196 (2022) 123–138, <https://doi.org/10.1016/j.actaastro.2022.04.002>.
- [73] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: 3rd International Conference for Learning Representations, 2015, <https://doi.org/10.2514/6.2018-2100>. San Diego, CA, USA.
- [74] M. Wilde, C. Clark, M. Romano, Historical survey of kinematic and dynamic space-craft simulators for laboratory experimentation of on-orbit proximity maneuvers,

- Progress in Aerospace Sciences 110 (2019), <https://doi.org/10.1016/j.paerosci.2019.100552>.
- [75] M. Zwick, I. Huertas, L. Gerdes, G. Ortega, Orgl – ESA's test facility for approach and contact operations in orbital and planetary environments. International Symposium on Artificial Intelligence, Robotics and Automation in Space, Madrid, 2018.
 - [76] H. Krúger, S. Theil, Tron – hardware-in-the-loop test facility for lunar descent and landing optical navigation, in: IFAC-ACA 2010 Automatic Control in Aerospace, 2010.
 - [77] V. Dubanchet, B. Romero, K.N. Gregersen, H. Austad, K. Gancet, et al., EROSS Project — European Autonomous Robotic Vehicle for On-Orbit Servicing. I-SAIRAS Virtual Conference, 2020.
 - [78] M. Piccinin, S. Silvestrini, G. Zanotti, A. Brandonisio, P. Lunghi, M. Lavagna, ARGOS: calibrated facility for Image based Relative Navigation technologies on ground verification and testing, in: 72nd International Astronautical Congress (IAC 2021, 2021, pp. 1–11.
 - [79] S. Silvestrini, P. Lunghi, M. Piccinin, G. Zanotti, M. Lavagna, Experimental validation of synthetic training set for deep learning vision-based navigation systems for lunar landing, in: 71st International Astronautical Congress, IAC 2020, 2020, pp. 1–10.
 - [80] T.H. Park, S. Sharma, S. D'Amico, Towards Robust Learning-Based Pose Estimation of Noncooperative Spacecraft, AAS/AIAA Astrodynamics Specialist Conference, Portland, 2019.
 - [81] I. Ali, O. Suominen, A. Gotchev, E. Ruiz Morales, Methods for simultaneous robot-world-hand-eye calibration: a comparative study, Sensors 19 (2019) 2837.
 - [82] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, 2017, p. 10884, [https://doi.org/10.1016/S1364-6613\(99\)01331-5](https://doi.org/10.1016/S1364-6613(99)01331-5).
 - [83] L. Arora, A. Dutta, Reinforcement learning for sequential low-thrust orbit raising problem, AIAA Scitech 2020 Forum 1 (PartF(January)) (2020) 1–15, <https://doi.org/10.2514/6.2020-2186>.
 - [84] A. Brandonisio, Deep Reinforcement Learning to Enhance Fly-Around Guidance for Uncooperative Space Objects Smart Imaging, 2020.
 - [85] A. Brandonisio, Sensitivity analysis of adaptive guidance via deep reinforcement learning for uncooperative space, AAS/AIAA Astrodynamics Specialist Conference (2021) 1–20.
 - [86] L. Federici, B. Benedikter, A. Zavoli, Machine learning techniques for autonomous spacecraft guidance during proximity operations, AIAA Scitech 2021 Forum (2021) 1–18, <https://doi.org/10.2514/1.a35076>.
 - [87] B. Gaudet, R. Linares, R. Furfaro, Deep reinforcement learning for six degree-of-freedom planetary landing, Advances in Space Research 65 (7) (2020) 1723–1741, <https://doi.org/10.1016/j.asr.2019.12.030>.
 - [88] M.L. Greene, C. Riano-Rios, R. Bevilacqua, N.G. Fitz-Coy, W.E. Dixon, Approximate optimal orbit transfer of non-cooperative debris, AIAA Scitech 2020 Forum 1 (PartF(January)) (2020) 1–13, <https://doi.org/10.2514/6.2020-1823>.
 - [89] E.T. Jaynes, Information theory and statistical mechanics, Physics Reviews 106 (1957) 620–630.
 - [90] S. Levine, V. Koltun, Learning complex neural network policies with trajectory optimization, Proceedings of the 31st International Conference on Machine Learning 32 (2014) 829–837. http://machinelearning.wustl.edu/mlpapers/papers/icml2014c2_levine14%5Cnpapers3://publication/uuid/0A6E5CB6-EB7C-4E25-8A0A-57697C1224BD.
 - [91] C.J. Watkins, P. Dayan, Q-learning, Machine Learning 8 (3–4) (1992) 279–292.
 - [92] A. Brandonisio, Deep Reinforcement Learning to Enhance Fly-Around Guidance for Uncooperative Space Objects Smart Imaging, Politecnico di Milano, 2020.

- [93] V. Mnih, et al., Asynchronous methods for deep reinforcement learning, *Proceedings of the 33rd International Conference on Machine Learning*, PMLR 48 (2016) 1928–1937.
- [94] N.D. Ratliff, J.A. Bagnell, M.A. Zinkevich, Maximum margin planning, in: *23rd International Conference on Machine Learning*, 2006, pp. 729–736, <https://doi.org/10.1145/1143844.1143936>.
- [95] N.D. Ratliff, D. Silver, J.A. Bagnell, Learning to search: functional gradient techniques for imitation learning, *Autonomous Robots* 27 (1) (2009) 25–53, <https://doi.org/10.1007/s10514-009-9121-3>.
- [96] S. Silvestrini, M. Lavagna, Model-based reinforcement learning for distributed path planning, *Advanced Space Technologies for Robotics and Automation* (2019) 0–7.
- [97] S. Silvestrini, M. Lavagna, Relative trajectories identification in distributed spacecraft formation collision-free maneuvers using neural-reconstructed dynamics, *AIAA Scitech 2020 Forum* (2020) 1–14, <https://doi.org/10.2514/6.2020-1918>.
- [98] B. Taskar, C. Guestrin, D. Koller, Max-margin Markov networks, in: *Advances in Neural Information Processing Systems*, 2004.
- [99] P. Abbeel, A.Y. Ng, Apprenticeship learning via inverse reinforcement learning, *Twenty-First International Conference on Machine Learning-ICML 1* (2004), <https://doi.org/10.1145/1015330.1015430>.
- [100] S. Silvestrini, M. Lavagna, Inverse reinforcement learning for collision avoidance and trajectory prediction in distributed reconfigurations, in: *70th International Astronautical Congress*, IAC 2019, 2019, pp. 1–6.
- [101] W. Chu, S. Wu, Z. Wu, Y. Wang, Least square based ensemble deep learning for inertia tensor identification of combined spacecraft, *Aerospace Science and Technology* 106 (2020) 106189, <https://doi.org/10.1016/j.ast.2020.106189>.
- [102] W. Chu, S. Wu, X. He, Y. Liu, Deep learning-based inertia tensor identification of the combined spacecraft, *Journal of Aerospace Engineering* 234 (2) (2020) 1356–1366, <https://doi.org/10.1177/0954410020904555>.
- [103] F. Baldini, A. Anandkumar, R.M. Murray, Learning Pose Estimation for UAV Autonomous Navigation and Landing Using Visual-Inertial Sensor Data,” 2020 American Control Conference, 2020. URL, <http://arxiv.org/abs/1912.04527>.
- [104] R. Furfaro, I. Bloise, M. Orlandelli, P. Di Lizia, F. Topputo, R. Linares, Deep learning for autonomous lunar landing, in: *AAS/AIAA Astrodynamics Specialist Conference*, 2018.
- [105] S. Wang, R. Clark, H. Wen, N. Trigoni, DeepVO: towards end-to-end visual odometry with deep recurrent convolutional neural networks, *Proceedings-IEEE International Conference on Robotics and Automation* (2017) 2043–2050, <https://doi.org/10.1109/ICRA.2017.7989236>.
- [106] L. Downes, T.J. Steiner, J.P. How, Deep learning crater detection for lunar terrain relative navigation, *AIAA Scitech 2020 Forum* (2020) 1–12, <https://doi.org/10.2514/6.2020-1838>.
- [107] L.M. Downes, T.J. Steiner, J.P. How, Lunar terrain relative navigation using a convolutional neural network for visual crater detection, in: *2020 American Control Conference (ACC)*, 2020, pp. 4448–4453, <https://doi.org/10.23919/ACC45564.2020.9147595>.
- [108] S. Silvestrini, P. Lunghi, M. Piccinin, G. Zanotti, M. Lavagna, Artificial intelligence techniques in autonomous vision-based navigation system for lunar landing, in: *71st International Astronautical Congress*, 2020, pp. 12–14.
- [109] S. Silvestrini, M. Piccinin, G. Zanotti, A. Brandonisio, I. Bloise, L. Feruglio, M. Varile, Optical navigation for lunar landing based on convolutional neural network crater detector, *Aerospace Science and Technology* 123 (2022) 107503.

- [110] A. Silburt, M. Ali-Dib, C. Zhu, A. Jackson, D. Valencia, Y. Kissin, D. Tamayo, K. Menou, Lunar crater identification via deep learning, *Icarus* 317 (2019) 27–38, <https://doi.org/10.1016/j.icarus.2018.06.022>.
- [111] A. Brandonisio, M. Lavagna, D. Guzzetti, Reinforcement learning for uncooperative space objects smart imaging path-planning, *Journal of the Astronautical Sciences* (2021), <https://doi.org/10.1007/s40295-021-00288-7>. URL, <https://link.springer.com/10.1007/s40295-021-00288-7>.
- [112] G. Ciabatti, S. Daftary, R. Capobianco, Autonomous planetary landing via deep reinforcement learning and transfer learning, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (2021) 2031–2038, <https://doi.org/10.1109/CVPRW53098.2021.00231>.
- [113] V. Pesce, A.A. Agha-Mohammadi, M. Lavagna, Autonomous navigation & mapping of small bodies, *IEEE Aerospace Conference Proceedings* (2018) 1–10, <https://doi.org/10.1109/AERO.2018.8396797>.
- [114] M. Piccinin, P. Lunghi, M. Lavagna, Deep reinforcement learning-based policy for autonomous imaging planning of small celestial bodies mapping, *Aerospace Science and Technology* 120 (2022) 107224, <https://doi.org/10.1016/j.ast.2021.107224>.
- [115] D.M. Chan, A.A. Agha-Mohammadi, Autonomous imaging and mapping of small bodies using deep reinforcement learning, *IEEE Aerospace Conference Proceedings* (2019), <https://doi.org/10.1109/AERO.2019.8742147>.
- [116] L. Federici, B. Benedikter, A. Zavoli, Deep learning techniques for autonomous spacecraft guidance during proximity operations, *Journal of Spacecraft and Rockets* 58 (6) (2021) 1–18, <https://doi.org/10.2514/1.A35076>.
- [117] B. Gaudet, R. Linares, R. Furfarò, Adaptive guidance and integrated navigation with reinforcement meta-learning, *Acta Astronautica* 169 (2020) 180–190, <https://doi.org/10.1016/j.actaastro.2020.01.007>.
- [118] B. Gaudet, R. Linares, R. Furfarò, Six degree-of-freedom body-fixed hovering over unmapped asteroids via LIDAR altimetry and reinforcement meta-learning, *Acta Astronautica* 172 (February) (2020) 90–99, <https://doi.org/10.1016/j.actaastro.2020.03.026>.
- [119] K. Hovell, S. Ulrich, Deep reinforcement learning for spacecraft proximity operations guidance, *Journal of Spacecraft and Rockets* 58 (2) (2021) 254–264, <https://doi.org/10.2514/1.A34838>.
- [120] A. Scorsoglio, A. D'Ambrosio, L. Ghilardi, B. Gaudet, F. Curti, R. Furfarò, Image-based deep reinforcement meta-learning for autonomous lunar landing, *Journal of Spacecraft and Rockets* (May) (2021) 1–13, <https://doi.org/10.2514/1.a35072>.
- [121] S. Willis, D. Izzo, D. Hennes, Reinforcement learning for spacecraft maneuvering near small bodies, *AAS/AIAA Space Flight Mechanics Meeting* 158 (2016) 1351–1368.
- [122] H. Li, Q. Gao, Y. Dong, Y. Deng, Spacecraft relative trajectory planning based on meta-learning, *IEEE Transactions on Aerospace and Electronic Systems* 57 (5) (2021) 3118–3131, <https://doi.org/10.1109/TAES.2021.3071226>.
- [123] Versal AI Core Series VCK190 Evaluation Kit. <https://www.xilinx.com/products/boards-and-kits/vck190.html>.
- [124] Xilinx VITIS AI. Adaptable and Real-Time AI Inference Acceleration. <https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html>.
- [125] FINN Framework. <https://xilinx.github.io/finn/>.
- [126] DNNDK User Guide. https://www.xilinx.com/support/documentation/user_guides/ug1327-dnndk-user-guide.pdf.
- [127] M. Tipaldi, B. Bruenjes, Survey on fault detection, isolation, and recovery strategies in the space domain, *Journal of Aerospace Information Systems* 12 (2) (2015) 235–256.

- [128] X. Olive, FDI (R) for satellites: how to deal with high availability and robustness in the space domain? *International Journal of Applied Mathematics and Computer Science* 22 (2012) 99–107.
- [129] M. Tipaldi, B. Bruenjes, Spacecraft health monitoring and management systems, in: 2014 IEEE Metrology for Aerospace (MetroAeroSpace), 2014, pp. 68–72.
- [130] M. Tipaldi, L. Glielmo, A survey on model-based mission planning and execution for autonomous spacecraft, *IEEE Systems Journal* 12 (4) (2017) 3893–3905.
- [131] A. Jónsson, R.A. Morris, L. Pedersen, Autonomy in space: current capabilities and future challenge, *AI Magazine* 28 (4) (2007), 27–27.
- [132] ECSS-E-ST-70-11C Space Engineering—Space Segment Operability, European Cooperation for Space Standardization Standard, 2008.
- [133] ECSS-E-ST-70-41C – Telemetry and Telecommand Packet Utilization, European Cooperation for Space Standardization Standard, 2016.
- [134] A. Zolghadri, D. Henry, J. Cieslak, D. Efimov, P. Goupil, Fault Diagnosis and Fault-Tolerant Control and Guidance for Aerospace Vehicles, Springer, London, UK, 2014.
- [135] A. Zolghadri, Advanced model-based FDIR techniques for aerospace systems: today challenges and opportunities, *Progress in Aerospace Sciences* 53 (2012) 18–29.
- [136] D. Lakey, M. Eiblmaier, M. Denis, B.T. de Sousa, R. Porta, M. Shaw, T. Francisco, Multi-mission end-to-end OBCP configuration control, in: SpaceOps 2012 Conference, 2012.
- [137] G. Furano, et al., Towards the use of artificial intelligence on the edge in space systems: challenges and opportunities, *IEEE Aerospace and Electronic Systems Magazine* 35 (12) (2020) 44–56.
- [138] SAVOIR-FDIR Handbook, European Space Agency, 2019. <https://essr.esa.int>.
- [139] J. Marzat, H. Piet-Lahanier, F. Damongeot, E. Walter, Model-based fault diagnosis for aerospace systems: a survey, *Journal of Aerospace Engineering* 226 (10) (2012) 1329–1360.
- [140] I. Hwang, S. Kim, Y. Kim, C.E. Seah, A survey of fault detection, isolation, and reconfiguration methods, *IEEE Transactions on Control Systems Technology* 18 (3) (2010) 636–653.
- [141] A. Wander, R. Forstner, Innovative fault detection, isolation and recovery on-board spacecraft: study and implementation using cognitive automation, in: IEEE Conference on Control and Fault-Tolerant Systems (SysTol), 2013, pp. 336–341.
- [142] M. Tafazoli, A study of on-orbit spacecraft failures, *Acta Astronautica* 64 (2–3) (2009) 195–205.
- [143] L. Troiano, M. Tipaldi, A. Di Cerbo, M. Hoping, D. De Pasquale, B. Bruenjes, Satellite FDIR practices using timed failure propagation graphs, in: Proceedings of the International Astronautical Congress, IAC, 2012, pp. 8524–8531.
- [144] L.M. Fesq, Current fault management trends in NASA’s planetary spacecraft, in: 2009 IEEE Aerospace Conference, 2009, pp. 1–9.
- [145] A.M. Madni, M. Sievers, Model-based systems engineering: motivation, current status, and research opportunities, *Systems Engineering* 21 (3) (2018) 172–190.
- [146] B. Bittner, M. Bozzano, A. Cimatti, R. De Ferluc, M. Gario, A. Guiotto, Y. Yushtein, An integrated process for FDIR design in aerospace, in: 4th International Symposium on Model Based Safety Assessment, IMBSA, 2014, pp. 82–95.
- [147] M. Tipaldi, L. Feruglio, P. Denis, G. D’Angelo, On applying AI-driven flight data analysis for operational spacecraft model-based diagnostics, *Annual Reviews in Control* 49 (2020) 197–211.
- [148] J.A. Martínez-Heras, A. Donati, Enhanced telemetry monitoring with novelty detection, *AI Magazine* 35 (4) (2019) 37–46.

- [149] S. Jaekel, B. Scholz, Utilizing artificial intelligence to achieve a robust architecture for future robotic spacecraft, in: Proceedings of the IEEE Aerospace Conference, 2015, pp. 1–14.
- [150] N. Muscettola, P. Nayak, B. Pell, B. Wiliams, Remote agent: to boldly go where no AI system has gone before, *Artificial Intelligence* 103 (1–2) (1998) 5–47.
- [151] ECSS-E-ST-40C –Software, European Cooperation for Space Standardization Standard, 2009.
- [152] M. Tipaldi, C. Legendre, O. Koopmann, M. Ferraguto, R. Wenker, G. D’Angelo, Development strategies for the satellite flight software on-board Meteosat Third Generation, *Acta Astronautica* 145 (2018) 482–491.
- [153] K. Reinholtz, K. Patel, Testing autonomous systems for deep space exploration, *IEEE Aerospace and Electronic Systems Magazine* 23 (9) (2008) 22–27.
- [154] G. Brat, E. Denney, D. Giannakopoulou, J. Frank, A. Jonsson, Verification of autonomous systems for space applications, in: Proceedings of the IEEE Aerospace Conference, 2006, pp. 1–11.
- [155] V. Nardone, A. Santone, M. Tipaldi, D. Liuzza, L. Glielmo, Model checking techniques applied to satellite operational mode management, *IEEE Systems Journal* 13 (1) (2018) 1018–1029.
- [156] P. Van Wesel, A. Goodloe, Challenges in the Verification of Reinforcement Learning Algorithms, NASA Langley Research Center, Hampton, VA, United States), 2017. Technical Report.
- [157] P. Blacker, C.P. Bridges, S. Hadfield, Rapid prototyping of deep learning models on radiation hardened CPUs, in: Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems, AHS), 2019, pp. 25–32.
- [158] M. Zoppi, M. Tipaldi, A. Di Cerbo, Cross-model verification of the electrical power subsystem in space projects, *Measurement* 122 (2018) 473–483.
- [159] R. Patton, F. Uppal, S. Simani, B. Polle, Robust FDI applied to thruster faults of a satellite system, *Control Engineering Practice* 18 (9) (2010) 1093–1109.
- [160] A. Falcoz, D. Henry, A. Zolghadri A, Robust fault diagnosis for atmospheric re-entry vehicles: a case study, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 40 (5) (2010) 886–899 (2010).
- [161] R. Fonod, D. Henry, C. Charbonnel, E. Bornschlegl, D. Losa, S. Bennani, Robust FDI for fault-tolerant thrust allocation with application to spacecraft rendezvous, *Control Engineering Practice* 42 (2015) 12–27.
- [162] H. Alwi, C. Edwards, A. Marcos, FDI for a Mars orbiting satellite based on a sliding mode observer scheme, in: Proceedings of the IEEE Conference on Control and Fault-Tolerant Systems, SysTol’10, 2010.
- [163] N. Tudoroiu, K. Khorasani, Satellite fault diagnosis using a bank of interacting Kalman filters, *IEEE Transactions on Aerospace and Electronic Systems* 43 (4) (2008) 1334–1350.
- [164] J.G. Meß, F. Dannemann, F. Greif, Techniques of artificial intelligence for space applications-A survey, in: Proceedings of the European Workshop on On-Board Data Processing, ESA OBPD2019), 2019.
- [165] H. Henna, H. Toubakh, M.R. Kafi, M. Sauyed-Mouchaweh, Towards fault-tolerant strategy in satellite attitude control systems: a review, *Proceedings of the Annual Conference of the PHM Society* 12 (1) (2020), 14–14.
- [166] L. Guo, N. Li, F. Jia, Y. Lei, J. Lin, A recurrent neural network based health indicator for remaining useful life prediction of bearings, *Neurocomputing* 240 (2017) 98–109.
- [167] Y. Huang, S. Li, J. Sun, Mars entry fault-tolerant control via neural network and structure adaptive model inversion, *Advances in Space Research* 63 (1) (2019) 557–571.

- [168] K. Hovell, S. Ulrich, On deep reinforcement learning for spacecraft guidance, in: Proceedings of the AIAA Scitech 2020 Forum, 2020.
- [169] D. Codetta-Raiteri, L. Portinale, Dynamic Bayesian networks for fault detection, identification, and recovery in autonomous spacecraft, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45 (1) (2014) 13–24.
- [170] NASA Small Spacecraft Systems Virtual Institute, “State-of-the-Art Small Spacecraft Technology”, October 2020.
- [171] Bryce and Space Technology, “Smallsats by the Numbers 2020”, 2020.
- [172] Space News, Cubist Movement, 2012.
- [173] E. Kulu, Nanosats Database, www.nanosats.eu.
- [174] H.-P. Chung, S.-H. Chang, C.-L. Hsieh, S.-L. Yang, Y.-H. Chen, Cubesat compatible fiber-optic gyroscope, in: Opto-Electronics and Communications Conference (OECC), 2020, pp. 1–3, 2020.
- [175] S. Douglas, J. Mitchell, S. Lee, Output drifting of vacuum packaged MEMS sensors due to room temperature helium exposure, *Journal of Sensor Technology* 3 (2013) 101–109.
- [176] S. Douglas, “Analysis of: MEMS Oscillator Sensitivity to Helium (Helium Kills iPhones) YouTube Video”, LinkedIn, 2019. <https://www.linkedin.com/pulse/analysis-mems-oscillator-sensitivity-helium-kills-iphones-doug-sparks/>.
- [177] Office of the Federal Register, Foreign Availability Determination Procedures and Criteria, 2015. Title 15 Part 768.7.
- [178] N.A.S.A./R. SciNews, “Cosmonauts Deploy Nanosatellites from outside the ISS”, 2017. <https://www.youtube.com/watch?v=-hutA7In7GA>.
- [179] D.T. Gerhardt, E. Scott, Passive magnetic attitude control for Cubesat spacecraft, in: 24th Annual AIAA/USU Conference on Small Satellites, 2010.
- [180] A. Lassakeur, C. Underwood, Magnetic cleanliness program on Cubesats for improved attitude stability, in: 2019 9th International Conference on Recent Advances in Space Technologies, RAST), 2019, pp. 123–129.
- [181] A. Alanazi, J. Straub, “Statistical Analysis of Cubesat Mission Failure”, 32nd Annual AIAA/USU Conference on Small Satellites, 2018.
- [182] T. Villela, C.A. Costa, A.M. Brandão, F.T. Bueno, Towards the thousandth CubeSat: a statistical overview, *International Journal of Aerospace Engineering* 2019 (2019). Article ID 5063145.
- [183] ECSS Secretariat, Space Engineering—Verification, ECSS-E-ST-10-02C, March 2009.
- [184] J. Shields, C. Pong, K. Lo, L. Jones, S. Mohan, C. Marom, I. McKinley, W. Wilson and L. Andrade, “Characterization of Cubesat reaction wheel assemblies”, *JoSS*, Vol. 6, No. 1, pp. 565–580
- [185] J.L. Schwartz, M.A. Peck, C.D. Hall, Historical review of air-bearing spacecraft simulators, *Journal of Guidance, Control, and Dynamics* 26 (4) (2003) 513–522.

Further reading

- [1] T. CubeSat Program, Cal poly SLO, Cubesat design specification, In Review 14 (2020).
- [2] ESA, Tailored ECSS Engineering Standards for In-Orbit Demonstration Cubesat Projects, TEC-SY/128/2013/SPD/RW, Issue 1 Rev 3, November 2016.
- [3] ESA, ESA Pointing Error Engineering Handbook, ESSB-HB-E-003, Issue 1 Rev. 0, July 2011.

This page intentionally left blank



Mathematical and geometrical rules

**Andrea Capannolo¹, Aureliano Rivolta², Andrea Colagrossi³,
Vincenzo Pesce³, Stefano Silvestrini¹**

¹Politecnico di Milano, Milan, Italy

²D-Orbit, Fino Mornasco, Italy

³Airbus D&S Advanced Studies, Toulouse, France

Mathematical and geometrical rules are the underlying basis for any engineering application. This section summarizes and reports some of the most useful mathematical and geometrical rules for guidance, navigation, and control (GNC) applications. In particular, matrix, vector, and quaternion algebra are discussed, together with some basic concepts of statistics. Finally, the expressions of the matrices to perform the rotation from the Earth-centered inertial (ECI) to the Earth-centered Earth-fixed (ECEF) are reported.



Matrix algebra

In mathematics, a matrix is a rectangular array of numbers arranged in rows and columns. A matrix is defined as:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix},$$

which is a matrix with m horizontal rows and n vertical column, and it is often referred to as a $m \times n$ matrix. Each element of a matrix is often denoted by a variable with two subscripts, i,j , indicating the row and the column, respectively. Note that a row vector is a $1 \times n$ matrix, and a column vector is a $m \times 1$ matrix.

Different fundamental operations are defined for matrices, and they compose the matrix algebra. The most basic ones are:

- *Matrix addition.* The sum $\mathbf{A} + \mathbf{B}$ of two $m \times n$ matrices \mathbf{A} and \mathbf{B} is calculated entry-wise:

$$(\mathbf{A} + \mathbf{B})_{i,j} = a_{ij} + b_{ij}.$$

The subtraction operation is analogously defined with the negative sign. Note that matrix addition/subtraction is defined only for matrices of the same size.

- *Scalar multiplication.* The product of a scalar c and a matrix \mathbf{A} is computed by multiplying every entry of \mathbf{A} by c :

$$c\mathbf{A} = [c a_{ij}].$$

- *Matrix transpose.* The transpose of an $m \times n$ matrix \mathbf{A} is the $n \times m$ matrix \mathbf{A}^T formed by turning rows into columns and vice versa:

$$\mathbf{A}^T = [a_{ji}].$$

This section is intended to be a preliminary introduction to matrix algebra. The reader is invited to deepen the topic in [1].

Square matrices

A matrix is said to be square if it has the same number of rows and columns (i.e., a matrix $n \times n$), and the entries a_{ii} form the main diagonal of a square matrix.

Some special square matrices can be defined:

- *Upper triangular matrix.* A matrix with all the elements below the main diagonal equal to zero.
- *Lower triangular matrix.* A matrix with all the elements above the main diagonal equal to zero.
- *Diagonal matrix.* A matrix with all the elements outside the main diagonal equal to zero.
- *Identity matrix.* A diagonal matrix in which all the diagonal elements are equal to 1. It is indicated with the symbol \mathbf{I}_n , with n as the matrix size (i.e., $n \times n$). It is called identity matrix because a matrix multiplied by the identity matrix is unchanged.
- *Symmetric matrix.* A matrix that is equal to its transpose: $\mathbf{A} = \mathbf{A}^T$.
- *Skew-symmetric matrix.* A matrix that is equal to the negative of its transpose: $\mathbf{A} = -\mathbf{A}^T$.

Matrix multiplication

Given two matrices \mathbf{A} and \mathbf{B} , the matrix \mathbf{C} is defined as multiplication of \mathbf{A} and \mathbf{B} if its components satisfy the following relation:

$$c_{ij} = \sum_k a_{ik} b_{kj} \quad (16.1)$$

where c, a, b represent the scalar elements of the matrices $\mathbf{C}, \mathbf{A}, \mathbf{B}$, respectively. From Eq. (16.1), it is observed that the single element c_{ij} is the result of a scalar product between a row of matrix \mathbf{A} , and a column of matrix \mathbf{B} , which must be of the same length. Hence, the number of columns of \mathbf{A} shall be equal to the number of rows of \mathbf{B} . Therefore, the multiplication of a matrix $m \times n$ and a matrix $n \times p$ produces a matrix of dimensions $m \times p$.

Properties

Differently from the scalar product, the matrix multiplication is a noncommutative operation Eq. (16.2):

$$\mathbf{AB} \neq \mathbf{BA} \quad (16.2)$$

However, it preserves the associativity and the distributivity properties, namely Eq. (16.3) and (16.4):

$$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC}) \quad (16.3)$$

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC} \quad (16.4)$$

The matrix multiplication can be transposed by transposing each matrix of the multiplication and inverting their order:

$$(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top \quad (16.5)$$

If \mathbf{A} and \mathbf{B} have complex elements, the complex conjugates of the multiplication are equivalent to the multiplication of complex conjugates of the single matrices:

$$(\mathbf{AB})^* = \mathbf{A}^* \mathbf{B}^* \quad (16.6)$$

Combining Eqs. (16.5) and (16.6), the conjugate transpose of the matrix multiplication satisfies the following equivalence:

$$(\mathbf{AB})^\ddagger = \mathbf{B}^\ddagger \mathbf{A}^\ddagger \quad (16.7)$$

where \ddagger indicates the conjugate transpose operation.

Additional properties characterize the matrix multiplication in the specific case of square matrices. As said, the identity matrix \mathbf{I}_n represent the neutral element of the multiplication, as:

$$\mathbf{A} \mathbf{I}_n = \mathbf{I}_n \mathbf{A} = \mathbf{A} \quad (16.8)$$

Powers of the matrix \mathbf{A} can be computed as sequential multiplications of itself, namely:

$$\mathbf{A}^k = \mathbf{A}\mathbf{A}\cdots\mathbf{A} \quad (16.9)$$

If \mathbf{A} and \mathbf{B} are square, the determinant can be defined for both. Such scalar quantity is a commutative parameter, in fact:

$$\det(\mathbf{AB}) = \det(\mathbf{BA}) = \det(\mathbf{A})\det(\mathbf{B}) = \det(\mathbf{B})\det(\mathbf{A}) \quad (16.10)$$

The same rule applies for the trace:

$$tr(\mathbf{AB}) = tr(\mathbf{BA}) \quad (16.11)$$

Matrix inversion

The inverse of a square matrix \mathbf{A} , labeled as \mathbf{A}^{-1} , is a matrix such that the following relation holds:

$$\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n \quad (16.12)$$

A necessary and sufficient condition for matrix \mathbf{A} to be invertible is to be nonsingular, i.e., to have a nonnull determinant:

$$\det(\mathbf{A}) \neq 0 \quad (16.13)$$

Analytical computation

The inverse matrix \mathbf{A}^{-1} can be computed through an analytical formula.

Given the matrix \mathbf{A} , the cofactor matrix $cof(\mathbf{A})$ can be computed such that:

$$cof(\mathbf{A})_{ij} = (-1)^{i+j} M_{ij} \quad (16.14)$$

where M_{ij} is the *minor* corresponding to the i -th row and j -th column of matrix \mathbf{A} , i.e., the determinant of matrix \mathbf{A} after i -th row and j -th column has been removed.

Then, the adjugate matrix is defined as the transpose of the cofactor matrix:

$$adj(\mathbf{A}) = cof(\mathbf{A})^\top \quad (16.15)$$

and the inverse matrix is finally expressed as Eq. (16.16) and (16.17):

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \text{adj}(\mathbf{A}) \quad (16.16)$$

The inverse of a matrix product can be computed, similarly to the transpose of Eq. (16.5), as:

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}. \quad (16.17)$$

Frobenius norm

The Frobenius norm of a matrix \mathbf{A} belongs to the group of *entry-wise* matrix norms. The general p -norm of \mathbf{A} reads:

$$\|\mathbf{A}\|_{p,p} = \left(\sum_{i=1}^n \sum_{j=1}^m |a_{ij}|^p \right)^{1/p} \quad (16.18)$$

where m and n are the two dimensions of the matrix \mathbf{A} .

The Frobenius norm, or simply the matrix norm, is then obtained from Eq. (16.18) by setting $p = 2$, and reads:

$$\|\mathbf{A}\|_{2,2} = \sqrt{\sum_{i=1}^n \sum_{j=1}^m |a_{ij}|^2} \quad (16.19)$$

The norm of Eq. (16.19) can be also expressed in alternate forms, namely Eq. (16.20):

$$\|\mathbf{A}\|_{2,2} = \sqrt{\text{tr}(\mathbf{A}^\dagger \mathbf{A})} = \sqrt{\sum_{i=1}^{\min\{n,m\}} \sigma_i^2(\mathbf{A})} \quad (16.20)$$

where \mathbf{A}^\dagger represents the conjugate transpose matrix of \mathbf{A} , and σ_i is its i -th singular value, obtained through the *singular value decomposition* (SVD), which is introduced in the following of this section.

Matrix rank

The rank of a matrix \mathbf{A} , $\text{rank}(\mathbf{A})$, represents the maximum number of linearly independent columns (and rows) of \mathbf{A} . If a matrix $(n \times n)$ has n linearly independent rows or columns, it is called a *full-rank* matrix. Rectangular matrices can have (at most) a rank equal to the smallest of its two dimensions.

A basic approach to compute the rank of a matrix is to reduce it to its *echelon form*, through sequential *Gaussian eliminations*. In the context of numerical computation, other approaches are adopted. Through SVD, it is possible to determine the rank of a matrix by identifying the number of non-null singular values. Such approach, although not the most efficient, represents the most numerically stable approach. Alternatively, other matrix decomposition can be leveraged to exploit a faster computation of the rank, such as the *rank-revealing QR factorization* [1].

Eigenvectors

The term *eigenpair* refers to the couple of a scalar eigenvector λ and its related eigenvector \mathbf{v} . These quantities refer to vector field transformation, but they can easily be associated to linear algebra if we consider a transformation on itself. Hence, the linear transformation that any square matrix can represent.

An eigenvector of a square matrix \mathbf{A} of dimensions $n \times n$ is:

$$\mathbf{Av}_i = \lambda_i \mathbf{v}_i \quad (16.21)$$

where the index i goes from 1 to n . This can be referred to as the eigen-equation. Should be noted that we implicitly took \mathbf{v}_i as a column vector; however, this can work also for row vectors as well, in this case, we would have:

$$\mathbf{u}_i \mathbf{A} = \kappa_i \mathbf{u}_i \quad (16.22)$$

where \mathbf{u}_i is a $1 \times n$ vector. In this case, the eigenpair is given by (κ_i, \mathbf{u}_i) and \mathbf{u}_i is referred to as the left eigenvector, while \mathbf{v}_i is the right eigenvector. This can be clearly associated with the left and right multiplication concept. For most applications, such as the GNC ones, the right eigenvectors are preferred. It should be noted that $\mathbf{u}_i \neq \mathbf{v}_i^T$; however, it is equal to the right eigenvector of \mathbf{A}^T .

The eigen-equation can be found in a different form:

$$(\mathbf{A} - \mathbf{I}_n \lambda_i) \mathbf{v}_i = 0 \quad (16.23)$$

From this form, we can deduce that the matrix $(\mathbf{A} - \mathbf{I}_n \lambda_i)$ must be singular. Hence, if we impose:

$$\det(\mathbf{A} - \mathbf{I}_n \lambda) = 0 \quad (16.24)$$

we obtain an equation in λ of order n , which is called characteristic equation or characteristic polynomial. Solving it allows to determine all the n

eigenvalues of \mathbf{A} . Being the equation an n order polynomial, it is also possible to write it as:

$$\det(\mathbf{A} - \mathbf{I}_n \lambda) = (\lambda_1 - \lambda)(\lambda_2 - \lambda)\dots(\lambda_i - \lambda)\dots(\lambda_n - \lambda) \quad (16.25)$$

where the eigenvalues are clearly visible. It should be noted that there is room for complex results, always in conjugate pairs if matrix \mathbf{A} is real. To each eigenvalue λ_i , it is possible to associate an eigenspace made of eigenvectors that satisfy the eigenequation, taking the null vector as well. If more than one nonnull eigenvectors can be associated to λ_i , we have a geometrical multiplicity case.

This shouldn't be a surprise: if we determine the eigenvalue of an $n \times n$ identity matrix, we know that all the eigenvalues are equal to 1, and each column of the matrix is an eigenvector. Namely, the identity matrix \mathbf{I}_n has n identical eigenvalues and n linearly independent eigenvectors associated to it. Any vector parallel to one of the n directions given by the columns still satisfies the eigenequation. Indeed, if we multiply by any scalar quantity the eigenequation, the result won't change. Hence, there are potentially infinite eigenvectors. If we want to identify one eigenpair, for the sake of understanding and for the computational use of the eigenvectors, we could select the only one with unitary norm. This, of course, doesn't apply to the case where a geometrical multiplicity happens.

A useful way to reduce the computation of eigenvalues could be to exploit some properties:

$$\begin{cases} \text{tr}(\mathbf{A}) = \sum_{i=1}^n a_{ii} = \sum_{i=1}^n \lambda_i \\ \det(\mathbf{A}) = \prod_{i=1}^n \lambda_i \end{cases} \quad (16.26)$$

Knowing all eigenvalues of a matrix can also be used to infer some characteristics of \mathbf{A} . For example, if even one eigenvalue is null, then the determinant of \mathbf{A} is null, the matrix is not invertible, and its rank is not maximum.

The eigenpairs of a matrix can also be used to diagonalize it. If we define a diagonal matrix, Λ , and a matrix, \mathbf{V} , which, respectively, collect all the n eigenvalues and n eigenvectors (i.e., one per eigenspace, preferably unitary), we can generalize the eigenequation as:

$$\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1} \quad (16.27)$$

That means that we can always find a transformation \mathbf{V} that links any matrix with a diagonal version composed by its eigenvalues.

If a matrix \mathbf{A} is Hermitian, or real and symmetric, all its eigenvalues are real. The relation between Hermitian matrices and their eigenvalue is further deepened also for positive-definite, semipositive-definite, and their negative counterparts, as the resulting eigenvalues would be $(\lambda > 0)$, $(\lambda \geq 0)$, $(\lambda < 0)$, and $(\lambda \leq 0)$, respectively.

It is also possible to link the eigenvalues of a matrix \mathbf{A} to the eigenvalue of closely related matrices. For example, any power \mathbf{A}^k matrix would have λ^k eigenvectors. Limitations may apply for singular matrices. Another link can be seen when adding an identity matrix, in fact, $\mathbf{A} + \gamma \mathbf{I}$ has eigenvalues that are all $\lambda + \gamma$. By extension, if we create a polynomial of \mathbf{A} , the resulting eigenvalues will have value equal to the polynomial result of the eigenvectors of \mathbf{A} .

Singular value decomposition

As we have seen before, it exists a decomposition of a square matrix \mathbf{A} such that $\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1}$ where \mathbf{V} is a collection of linearly independent eigenvectors and Λ is the diagonal matrix of the eigenvalues of \mathbf{A} . This is valid for square matrices, but we can find another decomposition for which this condition is not required.

The SVD is related to the polar decomposition and can be seen as a generalization of the factorization made using eigenvalues and eigenvectors. Given a $m \times n$ matrix, $\mathbf{A}_{m \times n}$, which can be complex, its SVD is:

$$\mathbf{A}_{m \times n} = \mathbf{S}_{m \times m} \mathbf{V}_{m \times n} \mathbf{D}_{n \times n}^\dagger \quad (16.28)$$

where $\mathbf{S}_{m \times m}$ and $\mathbf{D}_{n \times n}$ are singular, possibly complex, square matrices. $\mathbf{V}_{m \times n}$ is a diagonal rectangular matrix, and with \dagger , it has been indicated the Hermitian conjugate transposition. If $\mathbf{A}_{m \times n}$ is real, then $\mathbf{S}_{m \times m}$ and $\mathbf{D}_{n \times n}$ are also orthogonal and \dagger is replaced by a simple transposition. In literature, the notation $\mathbf{U}\Sigma\mathbf{V}^\dagger$ is also used to indicate the SVD.

The diagonal of $\mathbf{V}_{m \times n}$ is composed by the so-called singular values, and any null entries of this list would reduce by 1 the rank of $\mathbf{A}_{m \times n}$. The columns of $\mathbf{S}_{m \times m}$ and $\mathbf{D}_{n \times n}$ are, respectively, the left and right singular vectors of $\mathbf{A}_{m \times n}$. Geometrically speaking, the sequence of operations is a rotation or reflection $(\mathbf{D}_{n \times n}^\dagger)$, followed by a scaling ($\mathbf{V}_{m \times n}$) and ending with another rotation or reflection ($\mathbf{S}_{m \times m}$).

There is indeed a striking similarity with the eigenvalue decomposition; however, it should be noted that singular values and eigenvalues are not identical except for diagonal matrices. The definition of singular values, ν_i , is:

$$\begin{cases} \mathbf{A}_{m \times n} \mathbf{s}_i = \nu_i \mathbf{d}_i \\ \mathbf{A}_{m \times n}^\dagger \mathbf{d}_i = \nu_i \mathbf{s}_i \end{cases} \quad (16.29)$$

where \mathbf{s}_i and \mathbf{d}_i are the left and right singular vectors belonging to $\mathbf{S}_{m \times m}$ and $\mathbf{D}_{n \times n}$, and ν_i is the i -th singular values of $\mathbf{A}_{m \times n}$. The SVD can also be written in terms of outer product, meaning that we can also write:

$$\mathbf{A}_{m \times n} = \sum_{i=1}^n \nu_i \mathbf{s}_i \otimes \mathbf{d}_i, \quad (16.30)$$

which basically means that we can divide $\mathbf{A}_{m \times n}$ into the sum of matrices obtained by weighted outer products, whose bases are the vectors in $\mathbf{S}_{m \times m}$ and $\mathbf{D}_{n \times n}$.

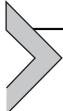
It should be remarked that there is no rule for the order of the singular values and the same applies for singular vectors. Hence, the decomposition is not unique unless some order is put in the values.

The SVD can be used also to facilitate the computation of the pseudoinverse of a matrix, in fact:

$$\mathbf{A}_{m \times n}^\dagger = (\mathbf{S}_{m \times m} \mathbf{V}_{m \times n} \mathbf{D}_{n \times n}^\dagger)^\dagger = \mathbf{D}_{n \times n} \mathbf{V}_{m \times n}^\dagger \mathbf{S}_{m \times m}^\dagger \quad (16.31)$$

where the pseudoinverse of a diagonal matrix is much easier to compute than the one of a full complete matrix.

Moreover, we can link the SVD to linear minimization. For example, $\min(\mathbf{Ax})$ with $\mathbf{x} = 1$ means that the solution is a right singular vector of \mathbf{A} , \mathbf{d}_i , such that its singular value ν_i is the smallest of all of them.



Vector identities

Vectors are geometric objects that have magnitude (or length) and direction, and they are used to represent geometrical or physical quantities. In matrix notation, a vector is a matrix with just one row (i.e., row vector) or one column (i.e., column vector). Vectors can be manipulated by different operators to obtain different results. Some common examples are the norm, the dot product, and the cross product. These have also an easily visualizable geometrical interpretation.

Vector norm

The L_2 norm of a vector \mathbf{v} is:

$$\| \mathbf{v}_2 \| = \sqrt{\mathbf{v}^T \mathbf{v}} \quad (16.32)$$

And is equivalent to the length of the vector. If we take a vector and multiply by the inverse of its norm, we will obtain a vector with unitary norm (i.e., length) often referred to as unit vector.

The operation under the square root operator is seen as the multiplication of the transpose of the vector by itself or, in general, it is the projection of a vector on itself. This projection operation is the dot product.

Dot product

The dot product between a vector \mathbf{v} and a vector \mathbf{u} is:

$$\mathbf{v} \cdot \mathbf{u} = \mathbf{v}^T \mathbf{u} = \sum_{i=1}^n v_i u_i \quad (16.33)$$

which represents the projection of one vector onto the other. It can be easily seen that $\mathbf{v} \cdot \mathbf{u} = \mathbf{u} \cdot \mathbf{v}$, the result is a scalar. The dot product exists only between vector of the same dimensionality. If \mathbf{u} is a unitary vector $\hat{\mathbf{u}}$, it means that the dot product gives the projection of \mathbf{v} along the direction $\hat{\mathbf{u}}$, meaning that the result is given by the length of \mathbf{v} and the cosine of the angle between the two directions. In other words:

$$\cos \vartheta = \frac{\mathbf{v} \cdot \mathbf{u}}{\| \mathbf{v} \| \| \mathbf{u} \|} = \hat{\mathbf{v}} \cdot \hat{\mathbf{u}} \quad (16.34)$$

Hence, the dot product is connected to the angle between two unitary vectors.

The dot product relates to the angle between two directions; however, this angle is computed in the plane formed by the two vectors, if the vectors are orthogonal, the dot product would be null, in fact, if $\mathbf{v} \perp \mathbf{u}$, then $\mathbf{v}^T \mathbf{u} = 0$.

Cross product

The operation that instead moves out of the plane is the cross product. The cross product of \mathbf{u} and \mathbf{v} is:

$$\mathbf{k} = \mathbf{v} \times \mathbf{u} = [\mathbf{v}_x] \mathbf{u} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \mathbf{u} \quad (16.35)$$

As you can see, this is the cross product in \mathbb{R}_3 . The result is still a tridimensional vector that by definition is orthogonal to the first two: $\mathbf{k} \perp \mathbf{v}$ and $\mathbf{k} \perp \mathbf{u}$. Looking closer, it is also reasonable to expect that $\mathbf{v} \times \mathbf{v} = 0$ and, in general, that if $\mathbf{u} \parallel \mathbf{v}$, $\mathbf{v} \times \mathbf{u} = 0$. Hence:

$$(\mathbf{v} \times \mathbf{u}) \cdot \mathbf{v} = (\mathbf{v} \times \mathbf{u}) \cdot \mathbf{u} = 0 \quad (16.36)$$

It should be noted that unlike the dot product, the cross product is anticommutative, meaning that $\mathbf{v} \times \mathbf{u} = -\mathbf{u} \times \mathbf{v}$. In fact, if we just take $k_1 = v_2u_3 - v_3u_2$ and $k_1 = u_2v_3 - u_3v_2$, it is clear that are opposite in sign. This is not a surprise as $[\mathbf{v}_\times]$ is skew-symmetric; hence, $[\mathbf{v}_\times]^T = -[\mathbf{v}_\times]$. Using the matrix form, we can also state:

$$\mathbf{v} \times \mathbf{u} = [\mathbf{v}_\times]\mathbf{u} = -[\mathbf{u}_\times]\mathbf{v} = -\mathbf{u} \times \mathbf{v} \quad (16.37)$$

Another property of the cross product is that it is addition distributive, $\mathbf{v} \times (\mathbf{u} + \mathbf{k}) = \mathbf{v} \times \mathbf{u} + \mathbf{v} \times \mathbf{k}$, and indifferent to a scalar multiplication $\mu(\mathbf{v} \times \mathbf{u}) = (\mu\mathbf{v}) \times \mathbf{u} = \mathbf{v} \times (\mu\mathbf{u})$. If we apply a matrix transformation, we have:

$$(\mathbf{Av}) \times (\mathbf{Au}) = \det(\mathbf{A})(\mathbf{A})^{-T} \mathbf{v} \times \mathbf{u} \quad (16.38)$$

If \mathbf{A} is a rotation matrix (i.e., unitary determinant, with transpose and inverse coincide), then applying a rotation to both vectors would apply the same rotation to their cross product: $(\mathbf{Av}) \times (\mathbf{Au}) = \mathbf{A}(\mathbf{v} \times \mathbf{u})$.

The shape of the cross matrix is in line with the ijk multiplication rule commonly used in rotation representation. If we take three unit vectors in a tridimensional cartesian reference frame $(\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}})$, thus orthogonal, we have that $\hat{\mathbf{z}} = \hat{\mathbf{x}} \times \hat{\mathbf{y}}$ but also $\hat{\mathbf{y}} \times \hat{\mathbf{x}} = -\hat{\mathbf{z}}$ and so on.

Like for the dot product, also the cross product can relate to the angle between two vectors. In the orthogonal unit vector case mentioned above, the resulting vector will always have a unitary norm which is the highest we can have. Let us take an example and use $\mathbf{k} = \hat{\mathbf{x}} \times \hat{\mathbf{s}}$ where $\hat{\mathbf{s}} = \{\cos \vartheta \quad \sin \vartheta \quad 0\}^T$, which is a vector displaced from $\hat{\mathbf{x}}$ of an angle ϑ . The result is $\mathbf{k} = \{0 \quad 0 \quad \sin \vartheta\}^T = \hat{\mathbf{z}} \sin \vartheta$. As expected, if $\vartheta = \frac{\pi}{2}$, we would go back to the previous orthogonal case. In the general case:

$$\sin \vartheta \hat{\mathbf{k}} = \frac{\mathbf{v} \times \mathbf{u}}{\|\mathbf{v}\| \|\mathbf{u}\|} \quad (16.39)$$

Another geometrical interpretation of the cross product is that the norm of a cross product identifies the area of the parallelogram generated on the (\mathbf{v}, \mathbf{u}) plane by the two vectors.

Dot and cross product share also combined identities besides those shown above. For instance:

$$\| \mathbf{v} \times \mathbf{u} \|^2 = \| \mathbf{v} \|^2 \| \mathbf{u} \|^2 - (\mathbf{v} \cdot \mathbf{u})^2 \quad (16.40)$$

That can easily be verified using trigonometry and the relations mentioned above.

If we take three vectors \mathbf{a} , \mathbf{b} , and \mathbf{c} , we have:

$$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = \mathbf{b} \cdot (\mathbf{c} \times \mathbf{a}) = \mathbf{c} \cdot (\mathbf{a} \times \mathbf{b}), \quad (16.41)$$

thanks to the multiplication rule mentioned above. But also:

$$\mathbf{a} \times \mathbf{b} \times \mathbf{c} = \mathbf{b}(\mathbf{a} \cdot \mathbf{c}) - \mathbf{c}(\mathbf{a} \cdot \mathbf{b}), \quad (16.42)$$

or:

$$(\mathbf{a} \times \mathbf{b}) \times (\mathbf{a} \times \mathbf{c}) = (\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}))\mathbf{a}. \quad (16.43)$$

Outer product

Another useful vector operator is the outer product that is not bounded by size like the cross product. The outer product of two vectors \mathbf{u} and \mathbf{v} is a matrix whose elements are the multiplication of each element of \mathbf{u} and \mathbf{v} in such a way that the i,j -th component of the matrix is equal to $u_i v_j$.

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u} \mathbf{v}^T \quad (16.44)$$

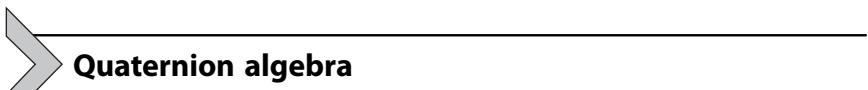
If the two vectors have the same dimensions, it follows that $\det(\mathbf{u} \otimes \mathbf{v}) = 0$, but in any case, the outer product matrix has rank 1. As one can see, the matrix version of the product is the inverse of the dot product; moreover, it is easy to see that:

$$\text{tr}(\mathbf{u} \otimes \mathbf{v}) = \mathbf{u} \cdot \mathbf{v} \quad (16.45)$$

This operation is not commutative, however:

$$(\mathbf{u} \otimes \mathbf{v}) = (\mathbf{v} \otimes \mathbf{u})^T. \quad (16.46)$$

The outer product is also associative and indifferent to scalar, as the cross product.



Quaternions have been introduced in [Chapter 5 – Attitude Dynamics](#), where they were closely linked to the attitude representation, but they were also necessary for the attitude determination of [Chapter 9 – Navigation](#). In these chapters, it has been used a specific multiplication rule; however, this is not unique as it will be clear in this section.

Quaternions have been introduced by William Rowan Hamilton in 1843 and are a mathematical tool that extend the complex algebra into three dimensions. Quaternions are extremely powerful to represent rotations and to deal with spacecraft attitude in GNC applications. A quaternion has a vector part, $\mathbf{q}_{1:3}$, and a scalar part, q_4 :

$$\mathbf{q} = \begin{Bmatrix} \mathbf{q}_{1:3} \\ q_4 \end{Bmatrix} = \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{Bmatrix}$$

It is worth noticing here that, depending on the textbook or the application, the quaternion definition may vary. Sometimes the scalar part is placed first, and it is noted as q_0 . Thus, there are two quaternion conventions: scalar first and scalar last. The latter is used in this book, with the scalar part in the fourth position. Unfortunately, the derived formulas and the associated mathematical operations are different according to the used convention. The reader is strongly encouraged to always verify and spend time on the quaternion convention that is in use before using any third-party function, library, or formula.

At this point, we can introduce the Hamiltonian left and right quaternion products, in matrix multiplication form:

$$\left\{ \begin{array}{l} [\mathbf{q}_\times^-] = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \\ [\mathbf{q}_\times^+] = \begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \end{array} \right. \quad (16.47)$$

and we link these matrices to the quaternion multiplication convention:

$$\mathbf{q}_c = \mathbf{q}_a \times \mathbf{q}_b = [\mathbf{q}_{a \times}^-] \mathbf{q}_b = [\mathbf{q}_{b \times}^+] \mathbf{q}_a \quad (16.48)$$

It should be noted that we choose to follow this convention, but we could have simply decided to use the opposite. This would have implied different mathematical relations.

From the matrix multiplication form, it follows clearly that, regardless of the multiplication order, we can always rearrange the operators. This comes in handy when we have to determine Jacobians for functions to be minimized, or when we analyze a specific rotational link in a long kinematic chain. For example:

$$\mathbf{q}_d = \mathbf{q}_a \times \mathbf{q}_b \times \mathbf{q}_c = [\mathbf{q}_{a \times}^-] [\mathbf{q}_{b \times}^-] \mathbf{q}_c = [\mathbf{q}_{a \times}^-] [\mathbf{q}_{c \times}^+] \mathbf{q}_b = [\mathbf{q}_{c \times}^+] [\mathbf{q}_{b \times}^+] \mathbf{q}_a \quad (16.49)$$

Here we have also used the properties of the quaternion multiplication, presented in matrix form:

$$\left\{ \begin{array}{l} [([\mathbf{q}_{a \times}^-] \mathbf{q}_b)_{\times}^-] = [\mathbf{q}_{a \times}^-] [\mathbf{q}_{b \times}^-] \\ [([\mathbf{q}_{a \times}^+] \mathbf{q}_b)_{\times}^+] = [\mathbf{q}_{a \times}^+] [\mathbf{q}_{b \times}^+] \\ [([\mathbf{q}_{a \times}^+] \mathbf{q}_b)_{\times}^-] = [\mathbf{q}_{b \times}^-] [\mathbf{q}_{a \times}^-] \\ [([\mathbf{q}_{a \times}^-] \mathbf{q}_b)_{\times}^+] = [\mathbf{q}_{b \times}^+] [\mathbf{q}_{a \times}^+] \end{array} \right. \quad (16.50)$$

So far, we haven't used the transposition operator on the matrices, and there is a reason for that: transposing one of the two multiplication matrix forms generates the counterrotation. Looking carefully at the off-diagonal components of the two multiplication matrices, being skew-symmetric, the transposition can be seen as an inversion of signs of the vector part of the quaternion. This means that it is equivalent to apply a multiplication for the conjugate quaternion:

$$\mathbf{q}^* = \left\{ \begin{array}{c} -\mathbf{q}_{13} \\ q_4 \end{array} \right\},$$

which represents the same rotation but in opposite verse. Hence, it is clear that:

$$[\mathbf{q}_{\times}^-]^T [\mathbf{q}_{\times}^-] = \mathbf{I}_4. \quad (16.51)$$

As a result, we can derive the matrix inverse as:

$$[\mathbf{q}_x^-]^{-1} = [\mathbf{q}_x^-]^T = [\mathbf{q}_x^{*-}], \quad (16.52)$$

where we have assumed the quaternion to have unit norm, as usually done in attitude applications. From these rules, we can also identify the quaternion division and further extend our capability to manipulate the rotation chains:

$$\mathbf{q}_d = [\mathbf{q}_{a\times}^-][\mathbf{q}_{b\times}^-]\mathbf{q}_c \rightarrow \mathbf{q}_c = [\mathbf{q}_{b\times}^-]^T[\mathbf{q}_{a\times}^-]^T\mathbf{q}_d. \quad (16.53)$$

Even in this case, it is possible to manipulate the chain to obtain different possibilities. It is, however, not possible to exchange \mathbf{q}_a and \mathbf{q}_d with the two operators used so far. New matrix forms need to be included:

$$\left\{ \begin{array}{l} [\mathbf{q}_x^{-T}] = \begin{bmatrix} -q_4 & q_3 & -q_2 & q_1 \\ -q_3 & -q_4 & q_1 & q_2 \\ q_2 & -q_1 & -q_4 & q_3 \\ q_1 & q_2 & q_3 & q_4 \end{bmatrix} \\ \\ [\mathbf{q}_x^{+T}] = \begin{bmatrix} -q_4 & -q_3 & q_2 & q_1 \\ q_3 & -q_4 & -q_1 & q_2 \\ -q_2 & q_1 & -q_4 & q_3 \\ q_1 & q_2 & q_3 & q_4 \end{bmatrix} \end{array} \right. \quad (16.54)$$

These matrices are no longer completely skew-symmetric, and the main diagonal has partially changed sign as well. The transposition operation is still maintained equal to the inversion for these matrices, moreover:

$$[\mathbf{q}_x^{-T}]^T = [\mathbf{q}_x^{+T}] \quad (16.55)$$

But also, the interesting relation:

$$[\mathbf{q}_x^{+T}]\mathbf{q} = [\mathbf{q}_x^{-T}]\mathbf{q} = \mathbf{q}_0 \quad (16.56)$$

where \mathbf{q}_0 is the null rotation vector: $\mathbf{q}_0 = \{ 0 \ 0 \ 0 \ 1 \}^T$.

Going back to our chain, we can now write:

$$\begin{aligned}\mathbf{q}_c &= [\mathbf{q}_{b \times}^-]^T [\mathbf{q}_{a \times}^-]^T \mathbf{q}_d = [\mathbf{q}_{b \times}^-]^T [\mathbf{q}_{d \times}^{-T}] \mathbf{q}_a = [\mathbf{q}_{d \times}^{-T}] [\mathbf{q}_{a \times}^-] \mathbf{q}_b \\ &= [\mathbf{q}_{d \times}^{-T}] [\mathbf{q}_{b \times}^+] \mathbf{q}_a\end{aligned}\quad (16.57)$$

From which also follows:

$$\left\{ \begin{array}{l} [\mathbf{q}_{a \times}^{-T}] = [\mathbf{q}_{b \times}^-] [\mathbf{q}_{a \times}^{-T}] [\mathbf{q}_{b \times}^+] \\ [\mathbf{q}_{a \times}^{+T}] = [\mathbf{q}_{b \times}^+] [\mathbf{q}_{a \times}^{+T}] [\mathbf{q}_{b \times}^-] \end{array} \right. \quad (16.58)$$

And:

$$\left\{ \begin{array}{l} [\mathbf{q}_{a \times}^{+T}] [\mathbf{q}_{b \times}^-]^T [\mathbf{q}_{a \times}^{-T}] = [\mathbf{q}_{b \times}^+] \\ [\mathbf{q}_{a \times}^{-T}] [\mathbf{q}_{b \times}^+]^T [\mathbf{q}_{a \times}^{+T}] = [\mathbf{q}_{b \times}^-] \end{array} \right. \quad (16.59)$$

Or:

$$\left\{ \begin{array}{l} [\mathbf{q}_{a \times}^{+T}] [\mathbf{q}_{b \times}^-]^T = [\mathbf{q}_{b \times}^+] [\mathbf{q}_{a \times}^{+T}] \\ [\mathbf{q}_{a \times}^{-T}] [\mathbf{q}_{b \times}^+]^T = [\mathbf{q}_{b \times}^-] [\mathbf{q}_{a \times}^{-T}] \end{array} \right. \quad (16.60)$$

That gives us, for example:

$$\mathbf{q}_d = [\mathbf{q}_{a \times}^-] [\mathbf{q}_{c \times}^+] \mathbf{q}_b = [\mathbf{q}_{c \times}^{-T}] [\mathbf{q}_{a \times}^+]^T [\mathbf{q}_{c \times}^{+T}] [\mathbf{q}_{c \times}^+] \mathbf{q}_b = [\mathbf{q}_{c \times}^{-T}] [\mathbf{q}_{a \times}^+]^T \mathbf{q}_b^* \quad (16.61)$$

where:

$$[\mathbf{q}_{a \times}^{+T}] [\mathbf{q}_{a \times}^+] = [\mathbf{q}_{a \times}^{-T}] [\mathbf{q}_{a \times}^-] = \text{conj}(\mathbf{q}_a) = \mathbf{q}_a^* \quad (16.62)$$

Is, of course, the matrix that transform any quaternion in its conjugate.

Quaternion from two directions

If the problem we are analyzing is not related to a long chain of rotation, but it is more closely related to attitude representation and estimation, what is the best way to determine a quaternion from two directions?

It is indeed possible to determine a quaternion such that the corresponding rotation move one vector onto the other. Since just a couple of vectors is not enough to determine attitude, there exist infinite rotations that can bring one vector onto the other, but it is possible to estimate quickly the shortest route.

First, using the cross product of the two vectors, it is possible to determine the Euler axis, meaning that the rotation we seek is perpendicular to both vectors and will bring one to the other. Then, we just need to estimate the angle between the two vectors. Mathematically, if $\mathbf{v}_2 = \mathbf{A}(\mathbf{q}_{21})\mathbf{v}_1$, we need to perform the following steps:

$$\begin{cases} e = \frac{\mathbf{v}_2 \times \mathbf{v}_1}{\| \mathbf{v}_2 \times \mathbf{v}_1 \|} \\ \vartheta = \cos^{-1}(\mathbf{v}_2 \cdot \mathbf{v}_1) \\ q_r = \begin{cases} \mathbf{e} \sin(\vartheta/2) \\ \cos(\vartheta/2) \end{cases} \end{cases} \quad (16.63)$$

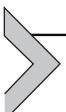
Note that using the cosine can give good results for most application; however, a more general procedure would see $\vartheta = \tan^{-1}\left(\frac{\|\mathbf{v}_2 \times \mathbf{v}_1\|}{\mathbf{v}_2 \cdot \mathbf{v}_1}\right) = \text{atan2}\left(\frac{\|\mathbf{v}_2 \times \mathbf{v}_1\|}{\|\mathbf{v}_2\| \|\mathbf{v}_1\|}, \frac{\mathbf{v}_2 \cdot \mathbf{v}_1}{\|\mathbf{v}_2\| \|\mathbf{v}_1\|}\right)$. Indeed, numerical stability should be considered as well.

It should be noted that, in general, \mathbf{q}_r does not coincide with \mathbf{q}_{21} , as any multiplication for a quaternion with Euler axis coincident with \mathbf{v}_2 will not compromise the projection of \mathbf{v}_1 onto \mathbf{v}_2 . This can be verified if we compute the relative quaternion:

$$\mathbf{q}_r = [\mathbf{q}_{r\times}^+]^T \mathbf{q}_{21} \quad (16.64)$$

Then, it will be easy to verify that $\mathbf{q}_{1:3r}/\sin(\cos^{-1}(q_{4r}))$ is equal to \mathbf{v}_2 . Please pay attention that this last procedure does not take into account the quaternion equivalence $\mathbf{q}_r = -\mathbf{q}_r$. A more robust way would be $\mathbf{q}_{1:3r} / \sin\left(\tan^{-1}\left(\frac{\|\mathbf{q}_{1:3r}\|}{q_{4r}}\right)\right)$ or, if one prefers computational efficiency over trigonometry, $\|\mathbf{q}_{1:3r}\| \text{sign}(q_{4r})$.

Quaternions are extremely important in many GNC applications, and the reader is invited to extend its knowledge about quaternions and quaternion algebra in [2].



Basics of statistics

Statistics is a collection of methods for collecting, displaying, analyzing, and drawing conclusions from data. It is fundamental for GNC applications since it may be applied both for processing the GNC functional data and for analyzing the results and the outputs during the GNC design, verification, testing, and operations. This brief section only introduces some basic concepts of statistics, which are also discussed in [Chapter 6 – Sensors](#). However, for a thorough study on the topic, the reader is invited to refer to the suggested reference [3,4].

Scalar statistics parameters

Given a random variable y , it is possible to define its *expected value* (or *mean*) as:

$$\mu = E(y) = \int_{-\infty}^{\infty} yf(y)dy \quad (16.65)$$

where $f(y)$ represents the *probability density function* of the random variable y .

The *variance* is defined as the expected value of the quadratic difference between the population of y and its mean value, namely:

$$\sigma^2 = var(y) = E(y - \mu)^2 \quad (16.66)$$

The square root of the variance is known as *standard deviation*:

$$\sigma = \sqrt{(E(y - \mu)^2)} \quad (16.67)$$

For two variables y_i and y_j , the *covariance* is defined as follows:

$$\sigma_{ij} = cov(y_i, y_j) = E[(y_i - \mu_i)(y_j - \mu_j)] \quad (16.68)$$

The covariance can be standardized by dividing it by the single standard deviation of the two variables. Such quantity is called *correlation* and reads:

$$\rho_{ij} = corr(y_i, y_j) = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \quad (16.69)$$

Vector and matrix forms of statistic quantities

Consider a random vector \mathbf{y} collecting N random variables y_i , such that:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \quad (16.70)$$

then, the *expected value* of the vector \mathbf{y} is the vector of expected values of its elements:

$$E(\mathbf{y}) = \begin{pmatrix} E(y_1) \\ \vdots \\ E(y_N) \end{pmatrix} = \boldsymbol{\mu} \quad (16.71)$$

The covariances of the y elements can be collected in a matrix form, also known as *covariance matrix*:

$$\boldsymbol{\Sigma} = cov(\mathbf{y}) = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1N} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{N1} & \sigma_{N2} & \cdots & \sigma_{NN} \end{bmatrix} \quad (16.72)$$

where σ_{ij} are the covariances, defined in Eq. (16.68), while $\sigma_{ii} = \sigma_i^2$ are the variances, as per Eq. (16.66). Notice that, due to the definition of the covariance, it results that $\sigma_{ij} = \sigma_{ji}$. Hence, the covariance matrix is symmetric. Furthermore, if the \mathbf{y} random variables are continuous and there are no linear relations among them, the covariance matrix is positive definite. In case linear relationships are present, the matrix is positive semidefinite.

From the expression of Eq. (16.69), it is possible to define the *correlation matrix*:

$$\mathbf{P} = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1N} \\ \rho_{21} & 1 & \cdots & \rho_{2N} \\ \vdots & \vdots & & \vdots \\ \rho_{N1} & \rho_{N2} & \cdots & 1 \end{bmatrix} \quad (16.73)$$

\mathbf{P} can be directly computed in matrix form from the covariance matrix $\boldsymbol{\Sigma}$ as:

$$\mathbf{P} = \mathbf{D}^{-1} \boldsymbol{\Sigma} \mathbf{D}^{-1} \quad (16.74)$$

with $D = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_N)$.

By analogy with Eq. (16.71), the *expected value* of a random matrix \mathbf{Y} is defined as the matrix of the expected values of its elements, namely:

$$E(\mathbf{Y}) = \begin{bmatrix} E(y_{11}) & E(y_{12}) & \cdots & E(y_{1N}) \\ E(y_{21}) & E(y_{22}) & \cdots & E(y_{2N}) \\ \vdots & \vdots & & \vdots \\ E(y_{N1}) & E(y_{N2}) & \cdots & E(y_{NN}) \end{bmatrix}. \quad (16.75)$$

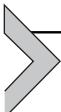
Hence, the covariance matrix Σ , defined in Eq. (16.72), can be also expressed as:

$$\Sigma = E[(\mathbf{y} - \boldsymbol{\mu})(\mathbf{y} - \boldsymbol{\mu})^T]. \quad (16.76)$$

It is possible to obtain a meaningful measure of the distance between \mathbf{y} and $\boldsymbol{\mu}$ by considering the variance and the covariance of each element y_i of the variables vector. Such measure is the *standardized distance*, also called *Mahalanobis distance* [5], and reads:

$$\text{Std. Dist.} = (\mathbf{y} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{y} - \boldsymbol{\mu}) \quad (16.77)$$

where the term Σ^{-1} standardizes the y_i variables, so that they will have means equal to 0 and variances equal to 1. In this way, they become uncorrelated.



ECI-ECEF transformation

The transformation between the ECI (*geocentric celestial reference frame - GCRF*) and the ECEF (*international terrestrial reference frame - ITRF*) reference frames is performed through a series of rotations that are known as *Earth orientation model*. An introductory discussion about this transformation is described in Chapter 2 — Reference Systems and Planetary Models.

The position, velocity, and acceleration conversions take the form of:

$$\mathbf{r}_{ECI} = \mathbf{P}(t) \mathbf{N}(t) \mathbf{R}(t) \mathbf{W}(t) \mathbf{r}_{ECEF} \quad (16.78)$$

$$\mathbf{v}_{ECI} = \mathbf{P}(t) \mathbf{N}(t) \mathbf{R}(t) (\mathbf{W}(t) \mathbf{v}_{ECEF} + \boldsymbol{\omega}_E \times \mathbf{r}_{PEF}) \quad (16.79)$$

$$\mathbf{a}_{ECI} = \mathbf{P}(t) \mathbf{N}(t) \mathbf{R}(t) [\mathbf{W}(t) \mathbf{a}_{ECEF} + \boldsymbol{\omega}_E \times (\boldsymbol{\omega}_E \times \mathbf{r}_{PEF}) + 2\boldsymbol{\omega}_E \times \mathbf{v}_{PEF}] \quad (16.80)$$

with:

$$\mathbf{r}_{PEF} = \mathbf{W}(t)\mathbf{r}_{ECEF} \quad \mathbf{v}_{PEF} = \mathbf{W}(t)\mathbf{v}_{ECEF}$$

where \mathbf{P} and \mathbf{N} are the precession-nutation matrices of date t , \mathbf{R} is the sidereal-rotation matrix of date t , \mathbf{W} is the polar motion matrix of date t , and ω_E represents the rotation rate of the Earth. Before expressing these rotation matrices, please refer to [Table 16.1](#) for a list of the relevant parameters to express the rotations, which are defined according to the IAU-76/FK5 Reduction. All the used coefficients are distributed by the International Earth Rotation and Reference Systems Service (IERS) [6].

Polar motion

The rotation matrix \mathbf{W} from the ECEF to the Pseudo-Earth Fixed frame is expressed as:

$$\mathbf{W} = \mathbf{R}_x(y_p) \mathbf{R}_y(x_p)$$

where x_p and y_p are the angular displacements of the true pole with respect to the International Reference Pole. Notice that the change in these values over one week is significant enough to require updates. Moreover, it is also

Table 16.1 List of relevant parameters for ECI/ECEF rotations.

x_p, y_p	Angular displacements of the true pole with respect to the international reference pole
$\theta_{GAST1982}$	Greenwich apparent sidereal time
θ_{GMST}	Greenwich mean sidereal time
T_{TT}	Terrestrial time
UTC	Coordinated universal Time
TAI	International atomic Time
$\Delta\Psi_{1980}$	Nutation in longitude
$\Delta\epsilon_{1980}$	Nutation in obliquity
$\delta\Delta\Psi_{1980}$	EOP corrections for the nutation in longitude
$\delta\Delta\epsilon_{1980}$	EOP corrections for the nutation in obliquity
$\bar{\epsilon}_{1980}$	Mean obliquity of the ecliptic
ζ, Θ, z	Precession angles
$M_{\mathbb{C}}$	Mean anomaly of the moon
M_{\odot}	Mean anomaly of the sun
$u_{M_{\mathbb{C}}}$	Mean argument of latitude of the moon
D_{\odot}	Mean elongation from the sun
$\Omega_{\mathbb{C}}$	RAAN of the mean lunar orbit measured on the ecliptic.

hard to accurately predict value over long periods because the pole motion has yet to be completely understood.

Sidereal time

The rotation matrix \mathbf{R} is the sidereal-rotation matrix of date:

$$\mathbf{R} = \mathbf{R}_z(-\theta_{GAST1982})$$

where $\theta_{GAST1982}$ is the Greenwich apparent sidereal time, computed from the mean sidereal time using the equation of the equinoxes:

$$\theta_{GAST} = \theta_{GMST} + Eq_{equinox}$$

$$Eq_{equinox} = \Delta\Psi_{1980} \cos(\bar{\epsilon}_{1980}) + 0.00264'' \sin(\Omega_{\mathbb{C}}) + 0.000063'' \sin(2\Omega_{\mathbb{C}}).$$

Nutation

The first step is to determine the mean obliquity of the ecliptic from:

$$\bar{\epsilon}_{1980} = 84381.448'' - 46.8150 T_{TT} - 0.00059 T_{TT}^2 + 0.001813 T_{TT}^3$$

T_{TT} is the Terrestrial Time, and it is related to other times as follows:

$$UTC = UT1 - \Delta UT1$$

$$TAI = UTC + \Delta AT$$

$$TT = TAI + 32.184s$$

$$T_{TT} = \frac{JD_{TT} - 2\ 451545}{36525}$$

where TAI is the *International Atomic Time* and ΔAT and $\Delta UT1$ are available from the EOP data. ΔAT is always an integer number and is constant until changed whereas $\Delta UT1$ changes continuously in time, as discussed in Chapter 2 – Reference Systems and Planetary Models.

Then the Delaunay parameters (or arguments), which account for the luni-solar nutation, are estimated with $r = 360^\circ$:

$$M_{\mathbb{C}} = 134.963\ 402\ 51^\circ + (1325r + 198.8675605)T_{TT} + 0.0088553T_{TT}^2 + 1.4343e^{-5}T_{TT}^3$$

$$M_{\odot} = 357.529\ 109\ 18^\circ + (99r + 359.050\ 291\ 1)T_{TT} - 0.0001537T_{TT}^2 \\ + 3.8e^{-8}T_{TT}^3$$

$$u_{M_{\mathbb{C}}} = 93.272\ 090\ 62^\circ + (1342r + 82.017\ 457\ 7)T_{TT} - 0.003542T_{TT}^2 - 2.88e^{-7}T_{TT}^3$$

$$\begin{aligned} D_{\odot} &= 297.850\ 195\ 47^\circ + (1236r + 307.111\ 446\ 9)T_{TT} \\ &\quad - 0.0017696T_{TT}^2 + 1.831e^{-6}T_{TT}^3 \end{aligned}$$

$$\Omega_{\mathbb{C}} = 125.044\ 555\ 01^\circ - (5r + 134.136\ 185\ 1)T_{TT} + 0.002\ 075\ 6T_{TT}^2 + 2.139e^{-6}T_{TT}^3$$

The nutation in longitude $\Delta\Psi_{1980}$ and in obliquity $\Delta\varepsilon_{1980}$ is evaluated through trigonometric series:

$$\Delta\Psi_{1980} = \sum_{i=1}^{106} (A_i + B_i T_{TT}) \sin(a_{p_i})$$

$$\Delta\varepsilon_{1980} = \sum_{i=1}^{106} (C_i + D_i T_{TT}) \cos(a_{p_i})$$

$$a_{p_i} = a_{n1i}M_{\mathbb{C}} + a_{n2i}M_{\odot} + a_{n3i}u_{M_{\mathbb{C}}} + a_{n4i}D_{\odot} + a_{n5i}\Omega_{\mathbb{C}}$$

To ensure compatibility with the GCRF, EOP corrections ($\delta\Delta\varepsilon_{1980}$, $\delta\Delta\Psi_{1980}$) are applied, obtaining:

$$\varepsilon_{1980} = \bar{\varepsilon}_{1980} + \Delta\varepsilon_{1980} + \delta\Delta\varepsilon_{1980}$$

$$\Delta\Psi_{1980} = \Delta\Psi_{1980} + \delta\Delta\Psi_{1980}$$

Finally, the transformation matrix is computed as:

$$\mathbf{N} = \mathbf{R}_x(-\bar{\varepsilon}_{1980})\mathbf{R}_z(\Delta\Psi_{1980})\mathbf{R}_x(\varepsilon_{1980})$$

Precession

The combined effects of precession are represented by the angles ζ , Θ , and z :

$$\zeta = 2306.2181''T_{TT} + 0.301\ 88T_{TT}^2 + 0.017\ 998T_{TT}^3$$

$$\Theta = 2004.3109''T_{TT} - 0.426\ 65T_{TT}^2 - 0.041\ 833T_{TT}^3$$

$$z = 2306.2181''T_{TT} + 1.094\ 68T_{TT}^2 + 0.018\ 203T_{TT}^3$$

The complete precession transformation matrix is:

$$\mathbf{P} = \mathbf{R}_z(\zeta)\mathbf{R}_y(-\Theta)\mathbf{R}_z(z)$$

References

- [1] D. Norman, D. Wolczuk, *Introduction to Linear Algebra for Science and Engineering*, Pearson Education, 2012.
- [2] F.L. Markley, J.L. Crassidis, *Fundamentals of Spacecraft Attitude Determination and Control*, Space Technology Library, Springer, New York, 2014.
- [3] J. Morrison, *Statistics for Engineers: An Introduction*, John Wiley & Sons, 2009.
- [4] W.J. DeCoursey, *Statistics and Probability for Engineering Applications with Microsoft Excel*, Newnes - Elsevier, 2003.
- [5] P.C. Mahalanobis, *On the Generalized Distance in Statistics*, National Institute of Science of India, 1936.
- [6] <https://www.iers.org/IERS/EN/DataProducts/EarthOrientationData/eop.html>.



Dynamical systems theory

Francesco Cavenago¹, Andrea Colagrossi², Stefano Silvestrini²,
Vincenzo Pesce³

¹Leonardo, Milan, Italy

²Politecnico di Milano, Milan, Italy

³Airbus D&S Advanced Studies, Toulouse, France

Modeling is typically the first step to be taken when the dynamics of a system is analyzed, and control and estimation algorithms need to be developed for spacecraft guidance, navigation, and control (GNC) applications. Several formulations exist, which can be more or less suitable depending on the situations and the scope of the system model. In this appendix, the primary modeling approaches used throughout this book are reviewed:

- State-space formulation.
- Discrete-time systems.
- Transfer functions.

This appendix chapter is intended to provide a brief overview of the fundamental concepts about dynamical systems theory, which is an essential topic for system modeling and GNC applications. The reader is strongly encouraged to extend knowledge about it in the suggested literature references [1–7].



State-space models

In state-space formulation, the time evolution of the system is described by the so-called state variables. Their name derives from the fact that they are able to characterize the condition, or state, of the system. Given the state at a certain instant of time and the inputs at the same instant, the time evolution of the system can be determined. In particular, this is formalized through a set of first-order differential equations in the following form:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, t), \\ \mathbf{x}(t_0) &= \mathbf{x}_0, \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}, \mathbf{u}, t),\end{aligned}\tag{17.1}$$

where \mathbf{f} and \mathbf{h} are $n \times 1$ and $p \times 1$ vector functions, sufficiently differentiable, \mathbf{x} is the $n \times 1$ state vector, \mathbf{u} is a $m \times 1$ vector collecting the input to the system, and \mathbf{y} is a $p \times 1$ vector gathering the output or measured signals. The dimension n of the state vector represents the order of the model. The system in Eq. (17.1) is time-varying, namely the dynamics functions explicitly depend on time. If this dependency is dropped, the model is said to be time-invariant. Another classification can be made according to the number of input and output. A system is called single-input-single-output (SISO) when $m = p = 1$, while it is called multi-input-multi-output (MIMO) if $m > 1$ and $p > 1$. The combination of the two previous alternatives is also possible (i.e., SIMO and MISO), and it is analogously defined.

A special class of systems, which is a subset of the general Eq. (17.1), is one of the linear systems. For this class, the state-space equations are expressed as follows:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu}, \\ \mathbf{x}(t_0) &= \mathbf{x}_0, \\ \mathbf{y} &= \mathbf{Cx} + \mathbf{Du},\end{aligned}\tag{17.2}$$

where \mathbf{A} is the $n \times n$ dynamics matrix, \mathbf{B} is the $n \times m$ control matrix, \mathbf{C} is the $p \times n$ sensor matrix, and \mathbf{D} is the $p \times m$ direct term. In linear time-varying system, these matrices depend on time, and the solution to Eq. (17.2) is given by:

$$\mathbf{x}(t) = \Phi(t, t_0)\mathbf{x}_0 + \int_{t_0}^t \Phi(t, \tau)\mathbf{B}(\tau)\mathbf{u}(\tau)d\tau,\tag{17.3}$$

where Φ is the state-transition matrix, whose properties are:

$$\begin{aligned}\Phi(t_0, t_0) &= \mathbf{I}, \\ \Phi(t_0, t) &= \Phi^{-1}(t, t_0), \\ \Phi(t_2, t_0) &= \Phi(t_2, t_1)\Phi(t_1, t_0), \\ \dot{\Phi}(t, t_0) &= \mathbf{A}\Phi(t, t_0).\end{aligned}\tag{17.4}$$

If the matrices \mathbf{A} , \mathbf{B} , \mathbf{C} , and \mathbf{D} do not depend on time, the system is called linear time-invariant (LTI), and the solution becomes:

$$\mathbf{x}(t) = e^{\mathbf{A}(t-t_0)}\mathbf{x}_0 + \int_{t_0}^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau,\tag{17.5}$$

where $e^{\mathbf{A}(t-t_0)}$ and $e^{\mathbf{A}(t-\tau)}$ are exponential matrices.

The conversion from a n -th order differential equation to a state-space model is possible. Let's consider the following SISO system:

$$\frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + a_{n-2} \frac{d^{n-2} y}{dt^{n-2}} + \dots + a_1 \frac{dy}{dt} + a_0 y = u. \quad (17.6)$$

A state-space vector can be defined as:

$$\mathbf{x} = \begin{bmatrix} d^{n-1}y/dt^{n-1} \\ d^{n-2}y/dt^{n-2} \\ \vdots \\ dy/dt \\ y \end{bmatrix}, \quad (17.7)$$

and, thus, the state-space model is:

$$\dot{\mathbf{x}} = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \cdots & -a_1 & a_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u, \quad (17.8)$$

$$y = [0 \ 0 \ 0 \ \cdots \ 1] \mathbf{x}.$$



Discrete-time systems

With the advent of digital computer, discrete-time systems have become more and more important for the implementation of estimation and control algorithms in GNC applications. Indeed, analog signals from the sensors are transformed to digital numbers through an analog-to-digital converter. Then, a computer is used to process the data for the GNC purposes. Finally, digital-to-analog converters transform back digital numbers to

analog signals to be provided to the system. A generic discrete-time system can be written as:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, t_k), \\ \mathbf{x}(t_0) &= \mathbf{x}_0, \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k, t_k),\end{aligned}\quad (17.9)$$

where k stands for the current instant of time.

The classifications and most of the concepts for continuous systems can be directly extended to discrete-time ones. Focusing on LTI systems, and assuming to use a zero-order hold system for sampling, the propagation of the state at the different sampling time can be derived by Eq. (17.5). Indeed, selecting as initial time a generic sampling instant kT , and $kT + T$ the subsequent instant, it is possible to write:

$$\mathbf{x}_{k+1} = e^{\mathbf{A}T} \mathbf{x}_k + \int_{kT}^{kT+T} e^{\mathbf{A}(T+\tau)} \mathbf{B} d\tau \mathbf{u}_k, \quad (17.10)$$

where T is the sampling period. Introducing $\beta = T + kT - \tau$, Eq. (17.10) becomes:

$$\mathbf{x}_{k+1} = e^{\mathbf{A}T} \mathbf{x}_k + \int_0^T e^{\mathbf{A}\beta} \mathbf{B} d\beta \mathbf{u}_k, \quad (17.11)$$

and, consequently, it turns out that the equation of the complete LTI discrete-time system takes a form similar to Eq. (17.2):

$$\begin{aligned}\mathbf{x}_{k+1} &= \tilde{\mathbf{A}} \mathbf{x}_k + \tilde{\mathbf{B}} \mathbf{u}_k, \\ \mathbf{x}(t_0) &= \mathbf{x}_0, \\ \mathbf{y}_k &= \mathbf{C} \mathbf{x}_k + \mathbf{D} \mathbf{u}_k,\end{aligned}\quad (17.12)$$

with

$$\tilde{\mathbf{A}} = e^{\mathbf{A}T}, \quad \tilde{\mathbf{B}} = \int_0^T e^{\mathbf{A}\beta} \mathbf{B} d\beta. \quad (17.13)$$

The matrices \mathbf{C} and \mathbf{D} remain unchanged with respect to the continuous-time case. Note that the derivation of Eqs. (17.10)–(17.11) is valid assuming that the input \mathbf{u}_k is constant in the sampling interval, as it

is when zero-order hold system is used (i.e., one of the most common sampling techniques).

As a final remark, it is recalled that an LTI discrete-time system is asymptotically stable if all the eigenvalues of the matrix $\tilde{\mathbf{A}}$ have modulus lower than 1.



Transfer functions

A very powerful approach to study a dynamic system is the frequency domain modeling, which is the foundation of classical control theory. The basic idea is to look at the response of the system to sinusoidal and exponential signals. Modeling in the frequency domain provides a deep insight of the behavior of the system, as shown in the section control design in [Chapter 10 - Control](#), and avoids the use of complex differential equations. Indeed, the relations are algebraic, which makes them easier to manipulate. For LTI systems, the mathematical tool used to describe the relation between an input and an output in the frequency domain is the transfer function. This can be determined in different ways, but the most common one is the ratio between the Laplace transforms of the output and the input when the initial conditions are zero. Before recalling the definition of the Laplace transform, it is noted that one of the aspects, which has made transfer functions very popular, is the possibility to represent them graphically through Bode and Nyquist plots, which provide particularly useful information about the properties of the dynamics, as explained in section control design in [Chapter 10 - Control](#).

Laplace transform is used to convert a function in a real variable, in the case of dynamical systems typically the time, to a function in a complex variable, s , usually called generalized frequency. Given a function $f(t)$ in the time domain, its Laplace transform is defined by:

$$\mathcal{L}(f(t)) = F(s) = \int_0^{\infty} e^{-st} f(t) dt, \quad \text{Re}(s) > s_0. \quad (17.14)$$

From this definition, some important properties of the Laplace transform can be derived, which are briefly recalled here:

- $\mathcal{L}(\alpha_1 f_1(t) + \alpha_2 f_2(t)) = \alpha_1 \mathcal{L}(f_1(t)) + \alpha_2 \mathcal{L}(f_2(t))$
- $\mathcal{L}(e^{at} f(t)) = F(s-a)$
- $\mathcal{L}(f(t-\tau)) = e^{-s\tau} F(s)$, for $\tau > 0$

- $\mathcal{L}\left(\int_0^t f(\tau) d\tau\right) = \frac{1}{s} F(s)$
- $\mathcal{L}\left(\frac{df(t)}{dt}\right) = sF(s) - f(0)$

Note that the last relation becomes simpler if the initial state is zero. Indeed, the Laplace transform of the time derivative can be obtained simply by multiplying the Laplace transform of $f(t)$ by s .

In order to show how to define a transfer function by using the Laplace transform, consider the following linear system:

$$\frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \dots + a_0 y = b_m \frac{d^m u}{dt^m} + b_{m-1} \frac{d^{m-1} u}{dt^{m-1}} + \dots + b_0 u, \quad (17.15)$$

where u and y are the input and output, respectively. The transfer function between u and y is computed by passing in the Laplace domain, i.e., by applying the relation for the time derivative, and assuming zero initial conditions. Consequently, Eq. (17.15) results in:

$$(s^n + a_{n-1}s^{n-1} + \dots + a_0)Y(s) = (b_m s^m + b_{m-1}s^{m-1} + \dots + b_0)U(s). \quad (17.16)$$

where $U(s)$ and $Y(s)$ are the Laplace transforms of the input and output, respectively. At this point, the transfer function $P(s)$ is simply the ratio between $U(s)$ and $Y(s)$:

$$P(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1}s^{m-1} + \dots + b_0}{s^n + a_{n-1}s^{n-1} + \dots + a_0}. \quad (17.17)$$

Transfer functions are characterized by three main features, which are the zero-frequency gain, the poles, and the zeros. Taking Eq. (17.17) as example, the gain is given by the magnitude of the transfer function at $s = 0$, i.e., $P(0) = \frac{b_0}{a_0}$, and it represents the ratio between the output and the input at steady state. The poles are the roots of the polynomial at the denominator. They are intrinsic characteristics of the system and determine its behavior (cfr. Table 2 in Section Control design in Chapter 10 - Control) and stability. In particular, the transfer function is stable if all its poles have negative real part. Finally, the zeros are the roots of the polynomial at the numerator. They affect the transient response of the system, and, if they have positive real part, they impose limitations to the maximum control

bandwidth that can be obtained (cfr. Section Limitations to control performance in [Chapter 10 - Control](#)). The location of the zeros in the complex plane depends on how sensors and actuators are integrated in the system, and thus modification to the design can move the zeros, potentially simplifying the control synthesis.

The transfer function between input and output can be found also starting from the state-space formulation. Consider [Eq. \(17.2\)](#) for the LTI case, moving to the Laplace domain, the following relations are obtained:

$$\begin{aligned} s\mathbf{X}(s) - \mathbf{x}(0) &= \mathbf{AX}(s) + \mathbf{BU}(s), \\ \mathbf{Y}(s) &= \mathbf{CX}(s) + \mathbf{DU}(s). \end{aligned} \quad (17.18)$$

Substituting $\mathbf{X}(s)$, isolated from the first row, in the second row, the output equation results to be:

$$\mathbf{Y}(s) = (\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D})\mathbf{U}(s) + \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0). \quad (17.19)$$

In the special case in which $\mathbf{x}(0) = 0$, [Eq. \(17.19\)](#) becomes:

$$\mathbf{Y}(s) = (\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D})\mathbf{U}(s), \quad (17.20)$$

and, thus, the transfer function $\mathbf{P}(s)$ is:

$$\mathbf{P}(s) = (\mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}). \quad (17.21)$$

For MIMO systems, $\mathbf{P}(s)$ is a matrix collecting all the transfer functions between each input and output. On the other hand, for SISO systems, $\mathbf{P}(s)$ becomes a scalar and can be written as $P(s) = \frac{Y(s)}{U(s)}$.

Transfer functions are often used in combination with block diagram representations to study dynamic systems, especially closed-loop systems. Exploiting block diagram algebra, it is possible to easily derive the output resulting from interconnected systems. To show this, consider the linear systems reported in [Fig. 17.1](#).

In [Fig. 17.1A](#), two systems, described by the transfer functions $P_1(s)$ and $P_2(s)$, are connected in series, and the relation between the input u and the output y can be derived as follows:

$$y = P_2v = P_2(P_1u). \quad (17.22)$$

Therefore, the overall transfer function of the series is $P = P_2P_1$, namely the product of the transfer functions of the two systems.

Consider now [Fig. 17.1B](#), which shows a parallel connection between the systems. The output of the interconnection results to be:

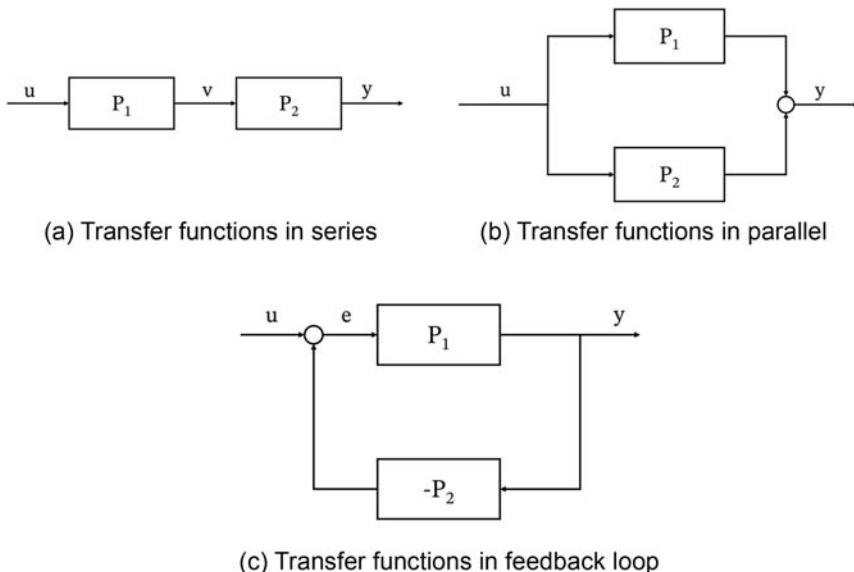


Figure 17.1 Interconnection of linear systems. In (A), two transfer functions are connected in series. In (B), two transfer functions are connected in parallel. In (C), two transfer functions create a feedback loop.

$$y = P_1 u + P_2 u = (P_1 + P_2) u, \quad (17.23)$$

and thus, the transfer function between the input and the output is simply the sum of the transfer functions, i.e., $P = P_1 + P_2$.

Finally, in Fig. 17.1C, a negative feedback system is reported. In this case, the output can be computed as:

$$y = P_1 e = P_1(u - P_2 y), \quad (17.24)$$

and consequently:

$$y = \frac{P_1}{1 + P_1 P_2} u. \quad (17.25)$$

From Eq. (17.25), it is evident that the overall transfer function is $P = \frac{P_1}{1 + P_1 P_2}$. These simple operations can be used to compute transfer functions in more complex systems.

To conclude this part, the transfer functions for some common LTI systems are listed:

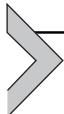
- Integrator: $\dot{y} = u \rightarrow P = \frac{1}{s}$
- Differentiator: $y = \dot{u} \rightarrow P = s$.
- Double integrator: $\ddot{y} = u \rightarrow P = \frac{1}{s^2}$.

- First-order system: $\dot{y} + ay = u \rightarrow P = \frac{1}{s+a}$.
- Second-order system: $\ddot{y} + 2\xi\omega_n\dot{y} + \omega_n^2y = u \rightarrow P = \frac{1}{s^2 + 2\xi\omega_n s + \omega_n^2}$.
- Time delay: $y(t) = u(t-\tau) \rightarrow P = e^{-s\tau}$.

References

- [1] N.S. Nise, Control Systems Engineering, John Wiley & Sons, Inc., 2011.
- [2] G.F. Franklin, D.J. Powell, A. Emami-Naeini, Feedback Control of Dynamic Systems, Pearson, 2015.
- [3] A. Tewari, Modern Control Design with MATLAB and SIMULINK, John Wiley & Sons, Inc., 2002.
- [4] K.J. Åström, R.M. Murray, Feedback Systems: An Introduction for Scientists and Engineers, second ed., Princeton University Press, 2021.
- [5] W.J. Rugh, Linear System Theory, Prentice-Hall, Inc., 1996.
- [6] T. Kailath, Linear Systems, vol. 156, Prentice-Hall, Englewood Cliffs, NJ, 1980.
- [7] H.K. Khalil, Nonlinear Systems, third ed., Patience Hall, 2002.

This page intentionally left blank



Autocoding best practices

Francesco Pace¹, Vincenzo Pesce², Andrea Colagrossi³,
Stefano Silvestrini³

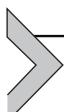
¹GMV Aerospace & Defence, Madrid, Spain

²Airbus D&S Advanced Studies, Toulouse, France

³Politecnico di Milano, Milan, Italy

Automated code generation or autocoding is more and more used in modern spacecraft guidance, navigation, and control (GNC), and it is almost becoming a baseline process for the GNC flight software (SW) application. The purpose of this appendix chapter is to provide modeling rules and guidelines to develop GNC models using MATLAB/Simulink in order to ensure the generated code being functionally correct, compliant with the existing standards as well as readable, reusable, and maintainable.

The content of this appendix chapter has to be intended together with the development, verification, and validation processes of the GNC subsystem, which are discussed in the core parts of this book. In particular, the reader is invited to combine the information contained in this appendix with those presented in [Chapter 12 - GNC Verification and Validation](#).



List of main architectural and implementation rules

The development described for the GNC system follows a SW life-cycle that uses autocoding techniques to generate code from MATLAB/Simulink models. Thus, the GNC model-based development shall follow several rules and guidelines for allowing the compatibility of the Simulink models with the autocoding process. Usually, autocoding rules are project-specific; however, some baseline guidelines can be defined derived from experience and autocoding standards [1]. It is strongly recommended to exercise the autocoding from the beginning of GNC SW development and then continue to test the autocoding on a regular basis.

Modeling rules for autocoding may be grouped into two categories:

- *Modeling architectural and design rules* defining the rules and guidelines that need to be followed at architectural and design level of the GNC subsystem models. These rules lead to efficiently port the code

to the validation and verification environments, and they guarantee architectural mapping.

- *Modeling implementation rules* defining the proper coding and implementation style to be followed by the Simulink GNC models. These rules prevent errors, language-specific pitfalls, nonoptimized statements, forbidden constructs, complexity restrictions, and readability in the generated code.

Architectural rules

For GNC modeling in a MATLAB/Simulink Functional Engineering Simulator (FES), the main architectural and design rules shall focus on the following aspects:

- *Architectural division between GNC on-board software (OBSW) models and real-world (RW) models.* The GNC receives the measurements from the RW, executes its algorithms, and provides commands back to the RW models closing the simulation loop. These two parts (OBSW and RW) must be clearly separated in the FES since only OBSW models will be autocoded for being embedded in a target space processor. It is recommended to keep the complete GNC system in a single block.
- *Clear definition of interfaces between the RW and the OBSW.* The interfaces between the RW and the OBSW shall be specified, and, in particular, the following signal characteristics must be defined (e.g., not inherited by Simulink):
 - Type
 - Dimension
 - Sample time

The following architectural rules shall be respected:

- Avoid algebraic loops inside OBSW chain.
- Include a unit delay block between the OBSW models and the RW.
- Set the sample time by subsystem properties of atomic block.
- The time used inside the GNC OBSW shall be passed as input via a “digital clock” block.
- Delays that simulate deterministic computation and transmission of measurements to GNC or commands from GNC to actuators shall be placed at top-level in a unique place.

Moreover, it is recommended to:

- Include models to simulate OBSW pre- and postprocessing operations (e.g., signal routing, rate transitions, pre- and postprocessing of signals, etc.) before and after GNC SW execution.
- Give univocal names to input and output signals.
- Organize the GNC modules according to their sample rate, grouping functions with the same sample time.
- Identify the main GNC functions and set them as atomic blocks to generate the corresponding C code functions. In this way, it is possible to guarantee the architectural mapping in the generated code.

Implementation rules

On the other hand, it is very important to clearly define low-level implementation rules that allow for a more standardized and error-free autocoding process. These guidelines can be divided into three main categories, depending on their applicability:

- *Mandatory*. In this category, there are all the essential guidelines to be followed. They are intended to ensure the correct behavior of the code generation process, without unexpected errors and corresponding to the origin model. If these rules are not followed, the autocoded block output can be inconsistent or inaccurate.
 - *Strongly recommended*. Rules that are recognized as good practice but not mandatory. Intended to ensure code reliability and representativeness of the original model. If not followed, the quality of the final code can decrease.
 - *Recommended*. Guidelines that are recommended to improve the model diagram but not critical for the final code behavior.
- In this chapter, only the main guidelines are detailed.

Mandatory

- Set the “Ensure data integrity during data transfer” parameter and “Ensure deterministic data transfer (maximum delay),” when rate-transition block is used (see Fig. 18.1).
- Avoid rate-transition blocks in atomic library subsystems.
- Avoid mask initialization commands. In fact, the variable initialized in this way are hardcoded in the autocoded model and not accessible in real time.
- Group and identify tunable and nontunable parameters for each mask present in the model (see Fig. 18.2). In the MATLAB environment, a tunable parameter is a parameter that can be changed without

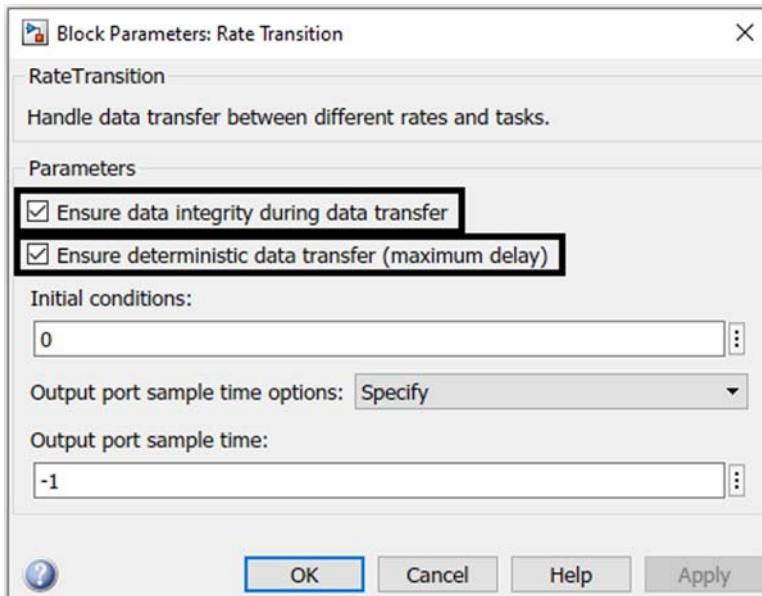


Figure 18.1 Rate-transition guidelines.

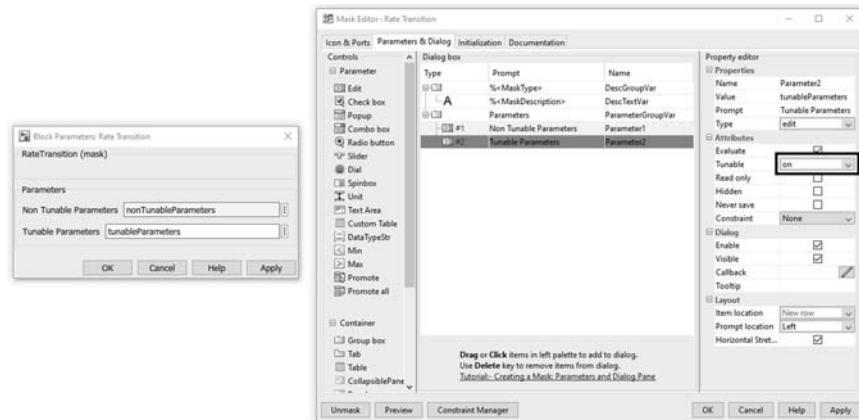


Figure 18.2 Tunable and nontunable parameters.

recompiling the model (or while running a simulation). Thus, tunable parameters can be accessed and updated in real time.

- Use simple mathematical expressions when tunable parameters are involved.

- Do not set as tunable parameters the delay or sample time in the Simulink blocks.
- Use external input to define index or initial conditions if tunable parameters are involved.
- Do not use callbacks in user-defined models.
- Protect all the division operation against the division by zero.
- Use only discrete models in the GNC SW.
- Define as atomic blocks the main functions of the GNC module. Use of atomic subsystems to define the functions and modules maximizes the architectural mapping between Simulink models and the produced C code. Avoid inherited sample times (see Fig. 18.3).
- Use univocal names for the atomic blocks defined in the model. Do not use reserved C words or operators or symbols for naming of the atomic

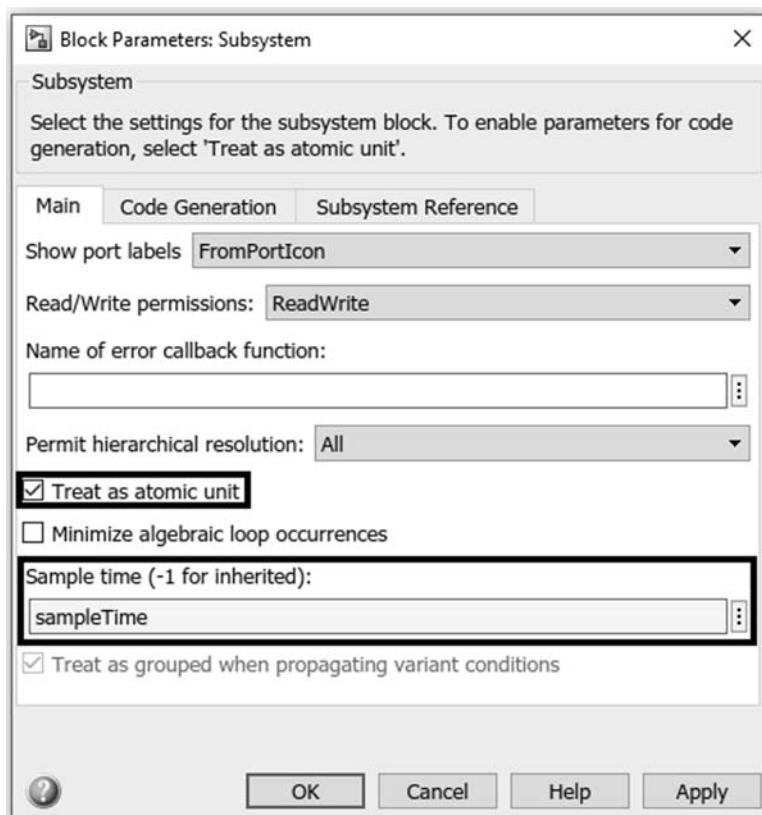


Figure 18.3 Atomic subsystem.

- blocks. Names of variables, in-port, out-port, bus objects, signals, and atomic blocks (as well as Stateflow variables and constant names) shall not use reserved C language words, operator, nor symbols.
- Use discrete data types (e.g., boolean, uint8) for parameters or variables that define discrete states (i.e., indexes, flags)
 - Define the initial value of blocks with internal states.
 - Do not use Global “From/GoTo” blocks between the GNC OBSW and the RW.
 - When using nested enabled subsystems and the same output signal is passed from a lower-level to a higher-level subsystem, the option “Output when disabled” of the out-port shall have the same settings (reset/held) throughout all the subsystems levels.
 - The output of a triggered subsystem shall not be used if system is not enabled. A switch outside the block should be implemented.
 - Avoid unconnected signals in the model. This can cause functional errors and problems with Embedded Coder. Use the “Terminator” block if necessary.
 - Include a function header for all the implemented functions.
 - Avoid performing logical operations with numerical operation blocks.
 - Set the library blocks (e.g., mathematical, GNC models, etc.) as atomic subsystems, with explicitly specified interfaces for data type and dimensions.
 - Set the “Criteria for passing first input” on the switch block as “ $u2 \sim = 0$ ” if the threshold signal (i.e., $u2$) is of Boolean type.
 - Define all the data types for the parameters that are not of double data type (e.g., index, flags, events, enumerated, etc.).
 - Do not use variable size signals in GNC OBSW.

Strongly recommended

- Avoid the definition of input/output signals of library models through a bus. In fact, each bus object shall be defined during the autocoding process, and this increases the effort in preparing the models.
- Set the option “Saturate on integer overflow” for all the blocks that provide this option (e.g., Data Type Conversion, Difference, etc.) as in Fig. 18.4.
- Avoid mux blocks and use vector concatenate instead, when possible.
- Use “From/GoTo” only within the same local view.

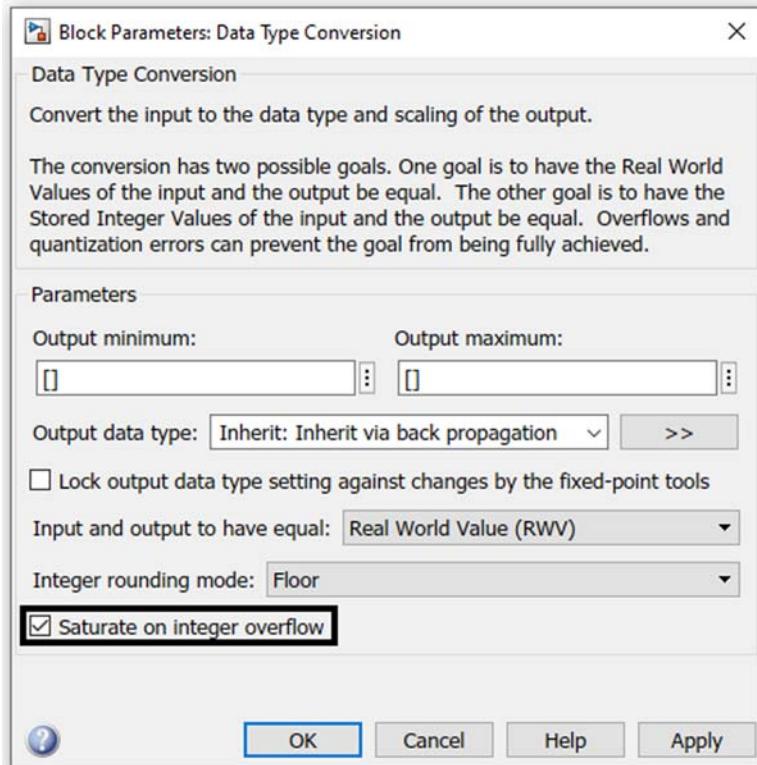


Figure 18.4 Saturate on integer overflow.

Recommended

- Adopt a naming convention for all the blocks of the model.
- Define the name of all the signals entering an atomic subsystem.
- Label all the signal lines for improved traceability.
- Minimize the use of “Math Function” blocks for exponential operations.
- Avoid the usage of C “S-Function” blocks. Correspondent source C code and Target Language Compiler files shall be used to generate OBSW code.

Configuration parameters setup

Another important aspect before starting the autocoding process is to correctly set the *Simulink Configuration Parameters*. In this paragraph, the most important suggested parameters corresponding to the main fields are summarized.

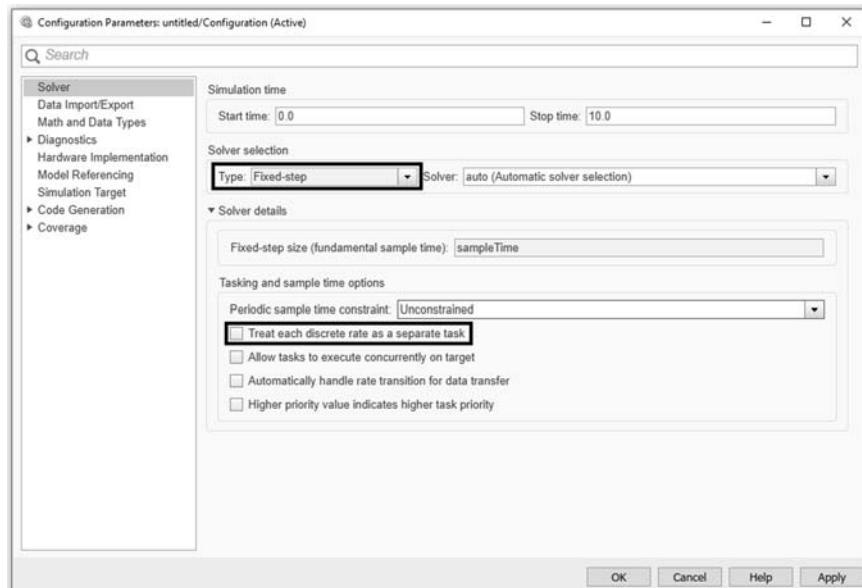


Figure 18.5 Solver.

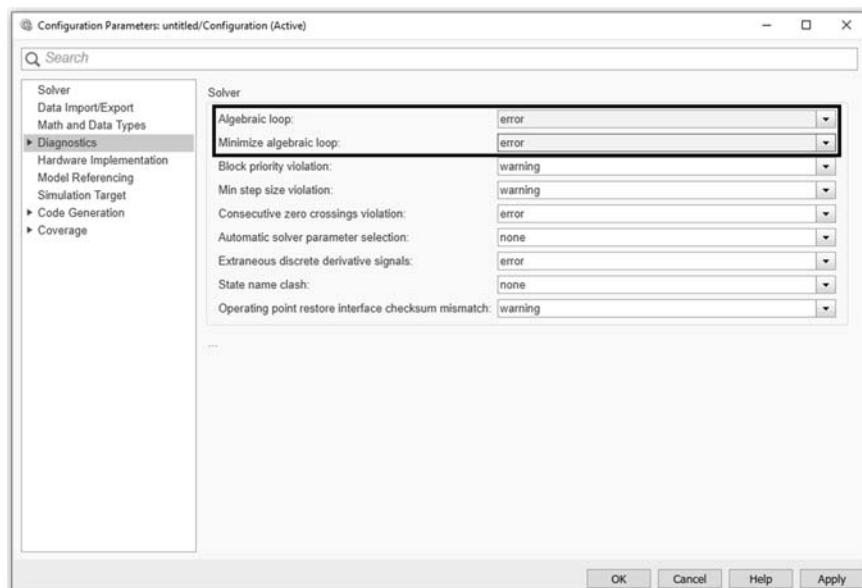


Figure 18.6 Diagnostics.

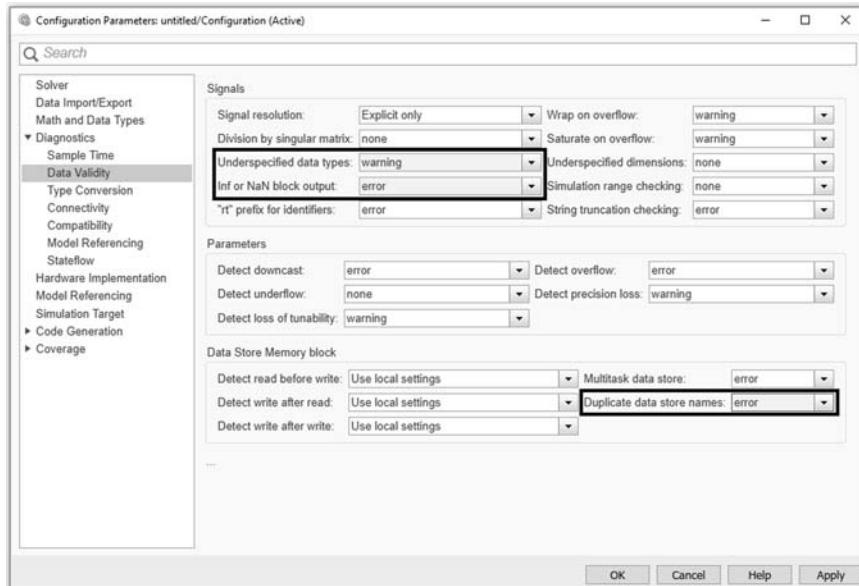


Figure 18.7 Data validity.

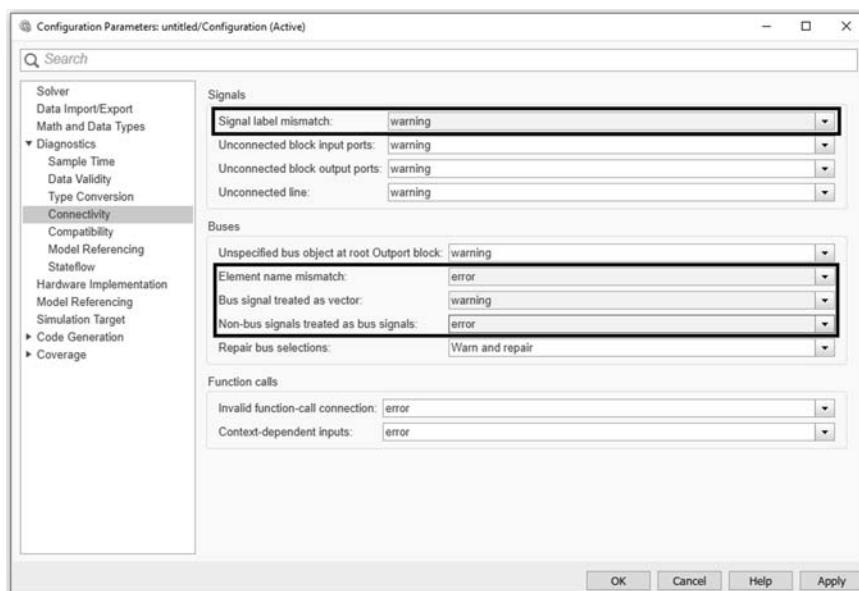


Figure 18.8 Connectivity.

- *Solver*. Use fixed-step solver and unselect “Treat each discrete rate as separate task” as in [Fig. 18.5](#).
- *Diagnostic*. Set parameters “Algebraic loop” and “Minimize algebraic loop” to error. See [Fig. 18.6](#).
- *Data validity*. Set “Underspecified data types” to warning, “Inf or NaN block output” to error, and “Duplicate data store names” to error as in [Fig. 18.7](#).
- *Connectivity*. Set “Signal label mismatch” to warning, “Element name mismatch” and “Nonbus signals treated as bus signals” to error, and “Bus signal treated as vector” to warning as in [Fig. 18.8](#).

Reference

- [1] Guidelines for Automatic Code Generation for AOCS/GNC Flight SW Handbook, ESA SAVOIR Working Group, 2021.

Index

Note: ‘Page numbers followed by “f” indicate figures and “t” indicates tables’.

A

- A3C. *See* Asynchronous Advantage Actor Critic (A3C)
Abramson model, 111–112
Absolute attitude navigation, 488–504
 complementary filter, 502–504
 Kalman filtering, 499–502
 triad, 489–491
 Wahba problem, 491–496
Absolute knowledge error (AKE), 620
Absolute orbit navigation, 470–488
 GNSS spacecraft navigation, 471–480
Absolute performance error (APE), 620,
 771
Absolute representation, 506–507
Acceleration-dependent bias, 313
Accelerometers, 306–310, 321–322
ACG. *See* Automated Code Generation (ACG)
ACS. *See* Attitude control system (ACS)
Activation function, 834–835
Active debris removal (ADR), 504
Active redundancy configuration, 24
Actor-critic network, 899–901
Actuation functions, 339–340
 magnetorquers, 372–373
 multiple reaction wheels, 360–363
 thrust management and, 347–353
Actuator (s), 16, 337, 544
 control action constraint, 612
 control moment gyros, 366–368
 faults, 342–344
 magnetorquers, 369–375
 misalignments, 433–434
 modeling for GNC, 338–344
 errors modeling, 341–342
 generic functional actuator model,
 341f
 generic performance actuator model,
 342f
 reaction wheels, 354–365
 thrusters, 344–354
Adaptive control, 600–605. *See also*
 Robust control
adaptive dynamical inversion, 603
additional parameters estimation, 604
convergence of parameters, 604
issues, 604–605
MRAC, 601–603
Adaptive Dynamical Inversion (ADI), 601,
 603
ADCS. *See* Attitude Determination and
 Control System (ADCS)
Additive EKF (AEKF), 499–500
Adjoint equations, 394
ADR. *See* Active debris removal (ADR)
Advantage Actor Critic (A2C), 900–901
Aerodynamic drag force, 245–246
Aerodynamic torque, 245–246
Aerospace optimization, 391
Agent algorithm, 823–824
AI. *See* Artificial intelligence (AI)
AIT. *See* Assembly Integration and Test
 (AIT)
AIV/AIT. *See* Assembly Integration and
 Verification/Assembly Integration
 and Test (AIV/AIT)
AKE. *See* Attitude knowledge error (AKE)
Albedo, 247–248
 emission, 93–94
Alexeev’s model, 99
AlexNet, 852–854
Allan variance (AVAR), 254, 317–321
 random walk noise and bias values for
 inertial sensors, 321t
Altimeters, 254, 330–334
 altimetry principles, 331–332
 model, 333–334
 radar and laser altimeters, 332–333
 radar *vs.* laser altimeter, 332t
Altimetry, 284
Altitude and Orbit Control system
 (AOCS), 14, 636, 820
AOCS/GNC algorithms, 656–663

- Altitude and Orbit Control system (AOCS)
(Continued)
architecture, 657–659
avionics delays, 659
GNC optimization, 661–662
modeling rules, 663
multirate, 659–660
requirements tracing, 661
tunable parameters, 660–661
- Ampère’s circuital law, 369
- Amplitude spectral density (ASD), 770
- Analog sensor, 292
- Analog sun sensors, 292–294
CSS, 292–294
FSS, 294
- Analytical redundancy, 24
- Angle random walk errors (ARW errors), 314
- Angular acceleration, 422
vector, 427
- Angular momentum, 230–231, 240–241
ellipsoid, 231–232
of rigid body, 230
- Angular rate vector, 221
- Angular velocity, 221–227, 422, 424
vector, 427
- ANNs. *See* Artificial neural networks (ANNs)
- AOCS. *See* Altitude and Orbit Control system (AOCS); Attitude and orbit control system (AOCS)
- APE. *See* Absolute performance error (APE)
- APF. *See* Artificial potential field (APF)
- API. *See* Application programming interface (API)
- Application programming interface (API), 925
- Application software level (ASW level), 657
- Application-specific integrated circuit (ASIC), 685, 693, 704–705
- ARM-based ASIC CPU, 703–704
- Architectural rules, 1018–1019
configuration parameters setup, 1023–1026
mandatory, 1019–1022
- atomic subsystem, 1021f
- rate-transition guidelines, 1020f
- tunable and nontunable parameters, 1020f
- recommended, 1023
- strongly recommended, 1022
- Argument of periapsis, 159
- Artificial intelligence (AI), 686, 819
AI-based architectures, 4
and navigation, 867–883
on-board processors, 923–925
in space, 821–866
AI, ML, DL, and ANN, 821–822
applications scenarios and AI challenges, 865–866
learning paradigms, 822–825
supervised learning, 826–843
types of artificial neural networks, 843–864
unsupervised learning, 825
use cases, 906–922
- Artificial neural networks (ANNs), 821–822, 835. *See also* Convolutional neural networks (CNNs)
architecture used in space dynamics, guidance, navigation, and control domain, 867t
cost function for multiclass classification, 838
model representation, 835–838
for multiclass classification, 834–843
parameter learning, 839–843
types of, 843–864
universal approximation theorem, 834–835
- Artificial potential field (APF), 780–783
active collision avoidance, 781–782
tracking controller, 782–783
- ARW errors. *See* Angle random walk errors (ARW errors)
- ARX model. *See* Autoregressive exogenous model (ARX model)
- Ascending node rotation, right ascension of, 66
- ASD. *See* Amplitude spectral density (ASD)

- ASIC. *See* Application-specific integrated circuit (ASIC)
- Assembly Integration and Test (AIT), 5, 936
- Assembly Integration and Verification/Assembly Integration and Test (AIT/AIV), 5
- ASW level. *See* Application software level (ASW level)
- Asymmetry scale factors, 267–268
- Asymptotically stable solution, 549
- Asynchronous Advantage Actor Critic (A3C), 900–901
- ATB. *See* Avionics Test Bench (ATB)
- Atmospheres, 245
- Atmospheric drag, 86–89
- Atmospheric models, 88, 99
- Atomic blocks, 1021
- Attitude and orbit control system (AOCS), 4–5, 382, 626, 715–730
control types, 722–726
evaluation of criticalities, 719–720
modes, 721–722
sensors selection and actuators sizing, 726–730
and subsystem architecture, 717–719
system-level trade-offs, 720–721
criteria matrix, 721t
- Attitude control system (ACS), 715–716, 742–773
control with magnetorquers, 757–758
control with reaction wheels, 753–756
detumbling, 743–745
effects of disturbances, 753
one-axis pointing, 745–749
robust attitude control of spacecraft with two rotating flexible solar arrays, 759–773
solar panels pointing, 759, 760f
three-axis pointing, 750–753
- Attitude Determination and Control System (ADCS), 14–15, 820
- Attitude dynamics, 9, 207–208, 227–248
attitude kinematics, 208–227
attitude stability, 235–239
dual spin dynamics, 239–241
environmental torques, 241–248
- inertia matrix, 227–229
- multibody spacecraft dynamics, 250–251
- relative attitude dynamics, 249–250
- rigid body dynamics, 230–235
- of satellite, 234–235
- three-body problem, 248–249
- Attitude guidance, 423–431
one-axis pointing, 423–426
two-axis pointing, 426–427
- Attitude kinematics, 208–227
attitude variation in time, 221–227
direction cosine matrix, 210–212
Euler angles, 212–215
Euler axis and angle, 215–217
quaternions, 217–221
- Attitude knowledge error (AKE), 726
- Attitude matrix, 232
- Attitude navigation, 488–489
- Attitude regulation example, 589–592
- Attitude sensors, 286–306. *See also* Orbit sensors; Electro-optical sensors; Global navigation satellite system sensors (GNSS sensors); Inertial sensors; Sun sensors
horizon sensors, 298–300
magnetometers, 287–290
performance comparison, 306
typical order of magnitude of performance for presented attitude sensor, 307t
star sensors, 300–306
sun sensors, 291–297
- Attitude sensors, 254
- Attitude stability, 235–239
attitude stability of torque-free motion, 237t
- Attitude variation in time, 221–227
angular velocity, 223–225
Euler angles kinematics, 225–226
quaternions kinematics, 226–227
- Augmented states, 458
- Augmented system model, 458
- Autocoding, 8
architectural and implementation rules, 1017–1026
- methodology, 668–671

- Automated Code Generation (ACG), 8, 1017
 Automatic attitude control systems, 12
 Automation, 31
 Autonomicity, 31
 Autonomous attitude determination, 302
 Autonomous navigation, 441–442
 Autonomous on-board sensor calibration, 796–800
 Autonomous segmentation, 508
 Autonomy, 31
 Autoregressive exogenous model (ARX model), 861
 Auxiliary satellite body coordinate systems, 62–63
AVAR. *See* Allan variance (AVAR)
 Average pooling, 851
 Avionics, 686
 on-board processing, 694–700
 spacecraft, 686–694
 Avionics Test Bench (ATB), 659
 Axis-angle, 388
 Azimuth, 59
- B**
- B-dot, 744–745, 746f
 Background randomization, 888
 Backpropagation, 839–843
 Backward propagation, 839
 Balancing, 955
 Ballistic coefficient, 87
 Barker’s equation, 149
 Bass–Gura formula, 576
 Batch estimation, 461–470
 dynamic effects, 463–464
 inclusion of priori information data, 466–467
 least squares, 462–463
 effect of observation errors, 464–465
 problems in batch orbit determination, 467–469
 square root information filter, 469–470
 U-D filter, 470
 Batch filtering, 443
 Batch filters, 461–462
 Batch learning, 842
 Batch orbit determination
 incorrect priori statistics and unmodeled parameters, 468
 numerical problems, 468–469
 presence of nonlinearities, 467
 problems in, 467–469
 Bayesian approach, 455
 Bayesian networks, 933
 Bayesian problem, 456
 Beam-limited sense, 331
 Behavioral models, 255
 BEIDOU, 278
 Bessel functions, 171
 Bias, 267
 bias–variance trade-off model, 830
 errors, 313
 estimator, 459
 instability, 951
 errors, 313
 repeatability, 313
 Binary classification
 cost function for, 834–843
 logistic regression for, 831–834
 Binary cross-entropy loss function, 833
 Biria–Russel–Vinti method, 201
 BLDC motors. *See* Brushless direct current motors (BLDC motors)
 Block redundant configuration, 24–25
 Bode plot, 556
 Bode’s integral formula, 581–582
 Boundary value problem (BVP), 739
 Brillouin sphere, 171
 Broucke STM, 201
 Brown noise, 270
 Brownian noise, 270
 Brushless direct current motors (BLDC motors), 953
 Bundle adjustment, 520–521
- C**
- Calibration techniques, 267–268
 Cameras, 254, 323–328, 507
 applicability, 325–327
 calibration, 519
 camera-based approaches, 325–327
 camera-to-mockup extrinsic calibration, 885–886
 design, 327–328

- intrinsic calibration, 885
matrix, 525
noise sources, 324t
relative motion of, 524–525
strengths and weaknesses, 328t
systems, 325–327
CAMs. *See* Collision Avoidance Maneuvers (CAMs)
CAN. *See* Controller Area Network (CAN)
Cannonball model, 90
Cardan angles, 213
Carrier phase, 471–472
 measurements, 279
Cartesian models, 198
Cartesian orbital elements mapping, 203–204
Cartesian space, 781–782
Cartesian state, 389
Cartography, 302
Categorical cross-entropy cost function, 838
Celestial Intermediate Origin approach, 64–65
Celestial Intermediate Pole (CIP), 64
Celestial pole offsets, 65
Center of Mass (CoM), 229, 338–339
 technique, 511–512
 variation, 115–116
Central limit theorem, 272
Central processing unit (CPU), 432–433
Centralized filters, 505–506
Centroid detection, 511–512
Centroding algorithm, 303–304
CFD. *See* Computational fluid dynamics (CFD)
Chain codes, 510
Charge-coupled device (CCD), 323
Chebyshev coefficients, 96
Chebyshev interpolation, 97
Chebyshev polynomials, 96
Chronometers, 3
CIP. *See* Celestial Intermediate Pole (CIP)
CIRA models, 88
Circular object detection, 511–512
 centroid detection, 511–512
 limb detection and fitting, 512
Circular orbits, 142, 148–149
 linearized equations of motion for nearly, 183–187
Circular relative orbit, 186–187
Circular Restricted Three-Body Problem (CRTBP), 162–164, 167–169
Cislunar space, 161, 248
Classical control theory, 569
Classical equinox-based transformation, 64–65
Classical least square estimator formula, 477
Classical spacecraft GNC, brief historical review of, 10–13
Clock bias, 483
Clock drift, 483
Clock errors, 482–483
Clock jitter, 483
Clohessy–Wiltshire model (C–W model), 183–184, 200, 203–204
Clohessy–Wiltshire–Hill equations, 571–572
Closed-loop
 dynamics, 580
 hand-eye calibration, 886–887
 response, 558
 stability, 548–549
 system, 730–731
CMGs. *See* Control moment gyros (CMGs)
CN. *See* Criticality number (CN)
CNNs. *See* Convolutional neural networks (CNNs)
Coarse Sun sensors (CSS), 292–294
Coefficients computation, 96
Cold gas propulsion systems, 345
Cold redundancy design, 24
Collinear Lagrangian Points, 164
Collision avoidance constraint, 612–613
Collision Avoidance Maneuvers (CAMs), 345
Collocation methods, 395
CoM. *See* Center of Mass (CoM)
Commercial off-the-shelf (COTS), 941
 components, 945–960
 GNSS receivers, 952–953
 inertial sensors, 950–951

- Commercial off-the-shelf (COTS)
(Continued)
- magnetic torquers, 956–960
 - magnetometers, 946–947
 - or custom, 945–946
 - reaction wheels, 953–955
 - star trackers, 949–950
 - sun sensors and earth sensors, 947–949
- Communication interfaces, 258
- Complementary filter, 502–504
- Complementary metal oxide semiconductor (CMOS), 323
- Complementary sensitivity function, 558
- Computational constraints, 533, 537
- Computational errors, 468
- Computational fluid dynamics (CFD), 111–112
- Computational power, 32–33
- Computer vision (CV), 824–825
- Concentric coplanar absolute orbit, 185–186
- Conics, geometrical classification of, 140–143
- Connectivity, 1025f, 1026
- Constant bias, 313
- Construct functional block diagram, 27
- Consumer grade, 315
- Continuation process, 166
- Continuous single-step Kalman filter, 913
- Contour-based shape representation, 510
- Control, 16, 543–545
 - budgets, 618–625
 - design, 545–592
 - basic terminology, 545–546
 - in frequency domain, 552–568
 - for nonlinear systems, 584–592
 - properties of feedback control, 546–547
 - in state space, 569–581
 - error, 558
 - gain, 579
 - horizon, 611–612
 - implementation best practices, 626–629
 - limitations to control performance, 581–584
 - objective and performance, 547–551
 - closed-loop stability, 548–549
- disturbance and measurement noise rejection, 550
- robustness to uncertainty, 551
- static and dynamic performance, 549–550
- review, 592–618
- Control moment gyros (CMGs), 338, 363, 366–368, 724–725
 - pyramidal 4 CMGs configuration, 367f
 - strengths and weaknesses, 368t
- Controllability, 551
- Controllability matrix, 575–576
- Controller, 544
- Controller Area Network (CAN), 258
- Convergence, 531–532
 - threshold, 531–532
- Convex optimization method, 811
- Convolution filters, 850–851
- Convolutional neural networks (CNNs), 819, 845–859
 - image classification networks, 852–854
 - image segmentation networks, 854–855
 - object detection network, 856–859
 - for planetary landing, 917–920
 - validation, 884–888
 - camera intrinsic calibration, 885
 - camera-to-mockup extrinsic calibration, 885–886
 - closed-loop hand-eye calibration, 886–887
 - error metrics, 887–888
- Coordinate reference systems, 53–63
 - geocentric earth-fixed coordinate system, 55–57
 - geocentric equatorial coordinate system, 54–55
 - heliocentric coordinate system, 53–54
 - lunar coordinate systems, 59–60
 - satellite-based coordinate systems, 61–62
 - three-body synodic and inertial coordinate systems, 60–61
 - LCROT, 60–61
 - topocentric coordinate systems, 57–59
 - topocentric equatorial, 57
 - topocentric horizon, 57–59
- Coordinate reference systems, 45
- Coordinate transformations, 45, 63–68

- ECI to ECEF, 64–65
ECI to PQW, 66–67
ECI to RSW, 67–68
Coordinated Universal Time (UTC), 70
Coordinates of pole, 65
Coordinates transformation, 193–195
Coplanar maneuvers, 733–735
Coriolis vibratory gyros, 311
Correction process, 166–167
Cosine detectors, 948
Cosmic velocities, 143–146
Cost function, 826–827, 833
 for binary classification, 834–843
 for multiclass classification, 838
Cotangential transfer, 412–414
COTS. *See* Commercial off-the-shelf (COTS)
Coulomb friction, 357
Coupled Ordinary Differential Equations, 107
Covariance
 computation, 879–882
 factorization of sequential filters, 536
 prediction, 448
Cowell’s formulation, 154–155
CPU. *See* Central processing unit (CPU)
Crater pattern matching algorithm, 812
Criticality number (CN), 28
Cross product operation, 38–39
Cross-coupling errors, 314–315
CRTBP. *See* Circular Restricted Three-Body Problem (CRTBP)
CSS. *See* Coarse Sun sensors (CSS)
CubeSats, 938–971
Cubic Hermite spline, 422
Customer requirements, 36–37
CV. *See* Computer vision (CV)
C–W model. *See* Clohessy–Wiltshire model (C–W model)
Cycle slip, 472
- D**
- D4PG algorithm. *See* Distributed distributional deep deterministic policy gradient algorithm (D4PG algorithm)
- Data arc, 487
- Data averaging, 318
Data clustering, 318
Data collection, 318
Data handling (DH), 686
Data loss fault, 275
Data processing unit (DPU), 692
Data validity, 1025f, 1026
Data-driven techniques for implementing FDIR systems in GNC applications, 931–934
Davenport method, 495
Davenport q-method, 493–495
Day-in-the-life test, 971
DCM. *See* Direction Cosine Matrix (DCM)
Decentralized filters, 505
Decision boundary, 832
Decision-making method, 634
Declination, 55
Deep learning (DL), 819, 821–822
Deep neural networks (DNNs), 821–822
Deep Q-learning network, 897–899
Deep Q-network (DQN), 898–899
Deep reinforcement learning
 algorithms, 897–903
 advantage actor-critic network, 899–901
 model-based reinforcement learning, 902–903
 PPO algorithm, 901–902
 Q-learning and deep Q-learning network, 897–899
for uncooperative objects fly around and planetary exploration, 920–922
meta-reinforcement learning, 921–922
Deep Space Network, 285
Degree of Freedom control (DoF control), 347
Delaunay elements, 390
Delay margin, 557
Delphi Deep Neural Network Development Kit (DNNDK), 925
Delta Differential One-way Range (DDOR), 484–485
Desaturation control, 755–756
Detection, 634

- Detection number (DN), 28
 Detumbling, 373–374, 743–745
 B-dot, 744–745
 classic, 743
 Development workflow for data-driven FDIR systems, 934–936
 Deviation angle, 114–115
 DH. *See* Data handling (DH)
 Dhystone million instructions per second (DMIP), 707–708
 Diagnostic parameters, 1024f, 1026
 Diagonal matrix, 984
 Digital elevation map (DEM), 813
 Digital sensor, 292
 Digital signal processors (DSPs), 685, 688–689
 Digital Sun sensors, 295–296
 Direct methods, 395, 739
 Direction Cosine Matrix (DCM), 208, 210–212, 387
 Discrete parameters modeling, example of, 103–106
 Discrete-time systems, 1009–1011
 Distributed distributional deep deterministic policy gradient algorithm (D4PG algorithm), 920–921
 Distributed parameters, 102
 example of distributed parameters modeling, 106–108
 Disturbance, 550
 rejection, 561–563
 DKE model. *See* Dynamics and kinematic environment model (DKE model)
 DL. *See* Deep learning (DL)
 DLR. *See* German Aerospace Agency (DLR)
 Doppler, 279, 283–284
 measurements, 472–473
 DP. *See* Dynamic programming (DP)
 Drag coefficient (c_D), 86
 Drag compensation control, 741
 Drift fault, 275
 Drift modulation transfer, 418
 Dry tuned-rotor gyros (DTGs), 310–311
 Dual spin dynamics, 239–241
 Dynamic attitude determination, 489, 499
 Dynamic imbalances, 121, 359
 Dynamic programming (DP), 895–896
 Dynamical equations, 232, 396
 Dynamical models, 507, 533–534, 537
 Dynamical system, 386
 discrete-time systems, 1009–1011
 state-space models, 1007–1009
 theory, 8
 transfer functions, 1011–1015
 Dynamical time, 69
 Dynamics acceleration reconstruction, 909–915
 Dynamics and kinematic environment model (DKE model), 659

E

- Earth Gravitational Models (EGMs), 47
 Earth Gravity Model 08 (EGM-08), 84–85
 Earth Gravity Model 96 (EGM-96), 84
 Earth magnetic field, 117
 Earth models, 45–53. *See also* Planetary models
 geoid and geopotential models, 52–53
 position representation, 48–52
 Earth orientation model, 64–65, 1002
 Earth Orientation Parameters (EOPs), 65
 Earth rotation angle, 64
 Earth sensors, 947–949
 Earth-based methods, 3
 Earth-centered Earth-fixed (ECEF), 52, 64–65, 282–283
 ECI to, 64–65
 reference frame conversion, 983
 spherical coordinates, 56–57
 Earth-centered inertial (ECI), 52, 282–283, 983
 ECI-ECEF transformation, 1002–1005
 nutation, 1004–1005
 polar motion, 1003–1004
 precession, 1005
 relevant parameters for ECI/ECEF rotations, 1003t
 sidereal time, 1004
 ECI-fixed attitudes, 943
 frame, 429
 to PQW, 66–67

- to RSW, 67–68
spherical coordinates, 55
- Earth’s equator, 54
Earth’s radius, 46
Earth’s rotational velocity, 47
East, North, Zenith (ENZ), 59
East longitude, 57
Eccentric anomaly, 150–151
Eccentricity, 157
vector, 134–135
- Eclipse, 92–93
- EGNOS, 278
- Eigenvectors, 988–990
- Eight-point algorithm, 524
- Electric propulsion (EP), 944
- Electrical ground support equipment (EGSE), 677
- Electrical jitter, 356
- Electro-optical sensors, 254, 322–329.
See also Inertial sensors; Attitude sensors; Global navigation satellite system sensors (GNSS sensors); Orbit sensors; Sun sensors
- cameras, 323–328
LIDAR, 328–329
- Electromagnetic disturbances, 117–118
- Electromagnetic induction, 342
- Elevation, 59
- Ellipsoidal model (ELL), 172–173
- Elliptic orbits, 150–151
- Elliptic Restricted Three-Body Problem (ERTBP), 162, 164–166, 169–170
- Embedded hard processor, 703
- End-to-end
approaches, 917
systems, 874
- Energetic analysis, 143–146
- Energy-matching condition, 196–198
- Entry, Descent, and Landing (EDL), 806
- Environment, 16–17
- Environmental torques, 241–248
aerodynamic torque, 245–246
gravity gradient torque, 241–244
magnetic torque, 244–245
solar radiation pressure torque, 246–248
- Ephemerides, 95–97
- Chebyshev interpolation, 97
Chebyshev polynomials, 96
coefficients computation, 96
error, 483
- Epipolar geometry, 521–523
- ϵ -greedy policy, 897–898
- Equations of motion, linearization of, 177–182
- Equatorial coordinate system (ECI), 57
- Equilateral Lagrangian Points, 164
- Equinoctial elements, 390
- Erratic fault, 275
- Error
bias, 267
function, 826–827
metrics, 887–888
misalignment and nonorthogonality, 273–274
modeling, 266–274
noise and random, 268–272
random errors with uniform distribution, 272
output saturation, temporal discretization, and latencies, 274
quantization, 272–273
scale factor, 267–268
- Essential matrix, 523
- Estimation error, 529–531
- Estimator gain matrix, 466
- ESTRACK, 285
- Euler angles, 212–217, 387–388
kinematics, 225–226
- Euler axis, 212, 215–217
- Euler dynamical equations, 248–249
- Euler equation, 232–235, 248–249, 388, 497, 743
- Euler’s equation, 423
- Euler’s theorem, 215
- European Cooperation for Space Standardization (ECSS), 97, 556–557, 632, 648, 705, 866 standards, 97
- European ECSS Space Segment Operability Standard, 35, 384
- European Improved Gravity model of Earth by New techniques 04 (EIGEN-GL04C), 84

- European Proximity Operations Simulator (EPOS), 883–884
- European Space Agency (ESA), 619, 632, 878–879
- European space missions, implementation in, 641–642
- European Space Research and Technology Centre (ESTEC), 883–884
- Experience replay, 899
- Exploding gradients, 841–842
- Exponential model, 88–89
- Extended Kalman filter (EKF), 450–452
- Extended normalization algorithm, 426
- Extended vector normalization, 427
- External perturbations, 77, 79–97. *See also* Internal perturbations
- atmospheric drag, 86–89
 - ephemerides, 95–97
 - gravitational models, 84–85
 - gravity field of central body, 80–84
 - magnetic field, 85–86
 - modeling guidelines, 97–100
 - atmospheric models, 99
 - gravity, 97–98
 - magnetic field, 98–99
 - solar radiation, 99–100
 - third-body perturbation, 100
- modeling guidelines, 77
- solar radiation pressure, 89–94
- third-body perturbation, 94–95
- Extrinsic camera calibration matrix, 518–519
- F**
- FAI. *See* Functional after integration (FAI)
- Failure Detection, Isolation and Recovery system (FDIR system), 631–632, 706–707, 716, 819
- challenges and next steps for industrial implementation of advanced FDIR systems, 936–938
- concept and functional architecture in GNC applications, 642–645
- data-driven techniques for implementing FDIR systems in GNC applications, 931–934
- development process and industrial practices, 637–639
- development workflow for data-driven FDIR systems, 934–936
- hierarchical architecture and operational concepts, 639–641, 639f
- implementation in European space missions, 641–642
- innovative techniques for highly autonomous FDIR in GNC applications, 820, 925–938
- model-based methods for implementing FDIR systems in GNC applications, 929–931
- in space missions, terms, and definitions, 633–637
- system, 17
 - conception, 27
 - evolution, 927–929
 - hierarchical architecture, 639–641
- terms and definitions, 636–637
- verification and validation approach, 642
- Failure detection and isolation (FDI), 794
- Failure detection method, 28
- Failure mode effect, 28
- Failure Mode Effects and Criticality Analysis (FMECA), 27, 636
- analysis approach, 27
- Fault detection and diagnosis (FDD), 929–930
- Fault Tree Analysis (FTA), 27, 638–639
- Fault-tolerant control (FTC), 929–930
- Fault-tolerant guidance (FTG), 929–930
- Feature engineering, 828–829
- Feature-matching approaches, 904–906
- Feedback control, properties of, 546–547
- Feedback linearization, 586–587
- Feedforward
 - design, 567–568
 - term, 579–580
 - torque, 423
- Ferromagnetic core torquers, 956
- Field programmable gate array (FPGA), 689–691, 702–703
- and hard-IP processor, 703
- multi-FPGA, 703
- single, 702

- softcore processor, 703
in space, 691–693
- Field-of-view scans (FOV scans), 298, 777
- Filter robustness, 529–531
- Filter smugness, 459
- Filtering process, 446–447
- Fine Sun sensors (FSS), 292, 294
- Finite elements methods (FEMs), 86, 103
- Finite State Machine (FSM), 30, 722
- Finite-horizon linear quadratic regulator, 597–598
- First-order differential equations, 1007–1008
- Flexibility of internal perturbations, 101–111
- Flexible appendages, 101–102, 251
- Flexible spacecraft, 759
- Flicker noise, 270
- Fluxgate magnetometers, 288–289
- Forced motion, 420–423
- Formation flying (FF), 445
- Forward propagation, 839
- Fourier transforms, 510
- François Viète’s Method, 149
- Frequency domain
control design in, 552–568
feedforward design, 567–568
loop-shaping design, 563–567
sensitivity functions, 557–563
stability and stability margins, 552–557
- Friction
jumps, 358
and microvibrations, 357–360
general kinetic friction model, 358f
reaction wheels waterfall plot, 360f
- Frobenius norm, 987
- Froude number, 111–112
- Full functional tests (FFTs), 677
- Fully neural dynamics learning, 907–909
- Functional actuator models, 340
- Functional after integration (FAI), 677
- Functional engineering simulator (FES), 657–658, 1018
- Functional models, 255
- Functional requirements, 432
- Functional sensor models, 256
- Fundamental matrix, 521
- estimation, 524
eight-point algorithm, 524
seven-point algorithm, 524
- ## G
- Gain
crossover frequency, 583
curve of open-loop transfer function, 565–566
margin, 555–556
scheduling, 586
- Galileo, 147, 278
- GAM STM, 201
- Gang of four, 558
- Gauss Variational Equation, 199
- Gaussian random variables, 263
- Gaussian Variation of Parameters (Gaussian VOPs), 155–159
- Gauss–Newton method, 827
- Gemini 6 spacecraft, 12–13
- General Kinetic Friction (GKF), 358
- Generalization test, 829–830
- Generalized frequency, 1011
- Generic discrete-time system, 1009–1010
- Geocentric Celestial Reference Frame (GCRF), 55
- Geocentric Earth-fixed coordinate system, 55–57
- Geocentric equatorial coordinate system, 54–55
- Geocentric latitude, 50, 57
- Geodesy, 46
- Geodetic latitude, 50
- Geoid models, 52–53
- Geoid’s undulation, 52
- Geometric boundary-based features, 510
- Geopotential models, 52–53
- Geostationary Earth orbit (GEO), 79–80, 280
- Geostationary East–West control, 741–742
- Geostationary North–South control, 741
- Geosynchronous/geostationary orbits (GEO), 147, 278
- German Aerospace Agency (DLR), 883–884
- Gim–Alfriend STM, 201

- Global methods, 509
Global navigation satellite system sensors (GNSS sensors), 47, 253, 277–283.
See also Orbit sensors; Attitude sensors; Electro-optical sensors; Sun sensors
accuracy, 281, 282t
basics, 277–279
error effects, 473–474
 Earth tidal effects, 474
 Ionospheric effects, 473
 Multipath effects, 474
 Relativistic effects, 474
 Tropospheric effects, 473–474
typical GNSS errors and uncertainty, 475t
GNSS-processed measurements, 478–479
ground-based orbit determination, 484–488
model, 282–283
multiconstellation GNSS receivers, 281–282
navigation approaches, 475–480
 precise orbit determination, 477–478
 precise point positioning, 476
 real-time navigation, 478–480
observables, 471–473
 carrier phase, 471–472
 Doppler measurements, 472–473
 pseudorange, 471
pulsar-based spacecraft navigation, 480–483
raw measurements, 479
receivers, 280–281, 952–953, 965
signals, 279–280
spacecraft navigation, 471–480
Global Navigation Satellite System–Inertial Navigation System (GNSS-INS), 716
integration, 479–480
 architecture, 480
 for on-board orbit determination, 800–802
Global navigation systems, 278
Global Positioning System (GPS), 57, 147, 278, 441–442
GLONASS, 147, 278
GMT. *See* Greenwich mean time (GMT)
GNC. *See* Guidance, Navigation, and Control (GNC)
GNC Rendezvous, Approach and Landing Simulator (GRALS), 883–884
GPUs. *See* Graphics processing units (GPUs)
Gradient descent (GD), 827, 839–843
Graphics processing units (GPUs), 685, 689, 870–872
Gravitational field, 82
Gravitational models, 84–85
Gravitational potential, 80
Gravity field of central body, 80–84
Gravity gradient torque, 241–244
Greenwich mean time (GMT), 70
Ground segment, 285
Ground station networks (GSNs), 277, 285
Ground track maintenance, 741
Ground truth (GT), 826
Ground-based control system, 32–35
Ground-based guidance, 382–385
Ground-based navigation, 443–445
Ground-based orbit determination, 484–488
 accuracy, 285
 ground-based OD strengths and weaknesses, 286t
 ground-based orbit determination accuracy, 286t
filter list example in standard orbit determination filter, 487t
techniques, 283–285
 ground segment, 285
 ground-based orbit determination accuracy, 285
 space segment, 285
Guidance, 15, 381–382
applications, 385–437
 design of guidance function, 431–431
 design process, 385–431
 attitude guidance, 423–431
 and control strategy, 779–789
 artificial potential field, 780–783

- impulsive, 779–780
model predictive control, 783–787
optimal control, 787–789
design of guidance function, 431–437
 architecture, 435–437
 function library, 437
 guidance modes, 434–435
 identification of guidance
 requirements, 431–434
design process, 385–431
general design approach, 385–386
guidance representations, 387–390
implementation best practices, 438
interpolation, 398–402
modes, 434–435
on-board *vs.* ground-based guidance, 382–385
optimization, 391–397
 classical formulation of optimal control problem, 392–395
 indirect methods *vs.* direct methods, 395
 trajectory optimization methods, 395–396
rendezvous guidance, 402–423
system, 382
 understanding dynamical system, 386
Guidance, Navigation, and Control (GNC), 3–4, 73–75, 124–126, 212, 253, 337, 382, 441–442, 633, 648–652, 715, 819, 983, 1007, 1017
AI
 in space, 821–866
 on-board processors, 923–925
 use cases, 906–922
algorithms, 685
anomalies, 26–28
architecture, 15–37
attitude control system, 742–773
COTS components, 945–960
design process, 18–20
effects
 on dynamics and, 108–111, 116–117
 on GNC chain, 532–533
innovative techniques for highly autonomous FDIR in GNC applications, 925–938
irregular solar system bodies fly around, 803–805
MIL test, 656–667
mode management, 22, 29–35
 automation, autonomy, and autonomicity, 31
 mode transition and finite state machine, 30
on-board *vs.* ground-based, 32–35, 35t–36t
model-based, 1017
notation rules, 38–42
 notation table, 39
on-board sensor processing, 791–802
orbital control system, 730–742
for planetary landing, 806–814
 planetary landing guidance, 806–814
preliminary design, 36–37
reinforcement learning, 890–906
relative, 775–791
requirements, 20–21
 operational requirements, 20
 performance requirements, 21
 verification requirements, 21
sensor modeling for, 254–274
 elements of metrology, 257–260
 errors modeling, 266–274
probability and stochastic processes, 261–265
random variables, 261–263
sensor calibration, 265–266
stochastic processes, 264–265
SIL/PIL test, 667–676
small satellites/CubeSats, 938–971
statistical methods, 652–655
subsystem design, 18–26, 1017
 GNC modes, 21–22
 mission life cycle, 25t
 mission phases, 25–26
 steps, 19t–20t
 system redundancy, 24–25
system, 77–78, 257
 example, 960–966
 GNSS receiver, 965
 inertial measurement unit, 963
 magnetometers, 964
 magnetorquers, 966
 reaction wheels, 965–966

- Guidance, Navigation, and Control (GNC)
(Continued)
- star trackers, 964
 - sun sensors, 963–964
 - terminology, 13–15
 - types of artificial neural networks, 843–864
 - validation of AI-based systems, 883–890
 - verification activities at MIL level, 663–667
 - verification and testing limitations, 966–971
 - hardware functional tests, 970–971
 - hardware performance tests, 969–970
 - hardware-in-the-loop, 971
 - software-in-the-loop, 968–969
- Gyroscope (GYRO), 306–309, 768
- model, 321–322
 - technology, 310–311
- H**
- H-infinity, 605–607
- $H\infty$ filter, 450
- Hall effect sensors, 356
- Hardover/bias fault, 275
- Hardware (HW), 635
- algorithm, 718
 - performance tests, 969–970
 - process, 820
- Hardware description language (HDL), 689–691
- Hardware-in-the-loop tests (HIL tests), 647, 677–682, 971
- for hardware verification, 678–680
 - for software verification, 680–682
- Hazard avoidance method, 814
- Health management systems, 631
- Heatmaps, 872–873
- Heliocentric coordinate system, 53–54
- Hermite spline, 400–401
- Herpolhode, 231–232
- Heuristic methods, 739
- Hidden layers, 836
- High Earth orbit (HEO), 281
- Hill–Clohessy–Wiltshire model (HCV), 200
- Homography, 513
- projective transformations, 514t
- Hopfield neural network (HNN), 862–863
- Horizon sensors, 298–300
- strengths and weaknesses, 300t
- Horizon-crossing indicators, 948
- Hot redundancy configuration, 24
- Hubble Space Telescope, 652
- Hybrid approaches, 917
- Hydrostatic equations, 87–88
- Hyperbolic orbits, 152
- Hyperbolic tangent, 837
- Hyperbolic trajectories, 142
- Hyperparameters, 830
- I**
- Ideal gas law, 87–88
- Identity matrix, 984
- Image classification networks, 852–854
- Image processing (IP), 527–528, 867–868
- segmentation, 507
 - global methods, 509
 - local methods, 508–509
 - and spacecraft navigation, 527–528
 - from subpixel to resolved object, 525–527
 - few tenths of pixels, 526
 - resolved object, 527
 - subpixel, 526
 - techniques, 507–528
 - 3D vision, 512–521
 - two-views geometry, 521–525
 - 2D shape representation, 509–512
- Image segmentation, 509–510
- networks, 854–855
- Implementation rules, 1019–1023
- configuration parameters setup, 1023–1026
 - mandatory, 1019–1022
 - recommended, 1023
 - strongly recommended, 1022
- Impulsive guidance, 779–780
- Impulsive maneuvers, 410–420, 731–733
- Impulsive shots, 187
- Impulsive trajectories, 410–420
- In-orbit test (IOT), 649, 682–683

- In-plane state transition matrix, 406
In-run stochastic variations, 317–318
Inclination, 157–158
 rotation, 66
Inclusion of priori information data, 466–467
Incremental learning, 842–843
Indirect methods, 395, 739
Inductive kickback/recirculation, 955
Industrial grade, 315
Inertia ellipsoid, 231–232
Inertia matrix, 227–229, 238–239
Inertia moments, 228
Inertia tensor, 228
Inertia wheel, 753–754
Inertial coordinate systems, 60–61
Inertial Measurement Unit (IMU), 306–307, 918, 963
Inertial Navigation System (INS), 479
Inertial sensors, 254, 306–322, 950–951.
 See also Sun sensors; Attitude sensors; Electro-optical sensors; Global navigation satellite system sensors (GNSS sensors); Orbit sensors
 Allan variance and statistical error representation, 317–321
 error sources, 312–315
 gyroscope model, 321–322
 performances, 315–316
 sensor grades and navigation performance after 1h of propagation, 317t
 technology, 308
Infinite-horizon linear quadratic regulator, 598
Infrared (IR), 93–94
 emission, 93–94
 radiation, 93–94
Initial attitude scattering, 653
Input layer, 836
Input range limits, 315
Integral action, 580–581
Integral quadratic constraints (IQC_s), 650
Integrals of motion, 134–136
 eccentricity vector, 134–135
 specific angular momentum, 134
 specific energy, 135–136
Inter-Integrated Circuit (I2C), 258
Interface and functional tests (IFT_s), 677
Internal geomagnetic field, 99
Internal Geomagnetic Field Model (IGRF), 47
Internal perturbations, 77–78, 100–122
 electromagnetic disturbances, 117–118
flexibility, 101–111
 effects on dynamics and GNC, 108–111
 example of discrete parameters modeling, 103–106
 example of distributed parameters modeling, 106–108
internal vibrations, 118–121
 reaction wheel jitter, 119–121
modeling guidelines, 78, 123–124
parasitic forces and torques due to plume impingement, 121–122
parasitic forces and torques during thrusters firing, 113–117
 center of mass variation, 115–116
 deviation angle, 114–115
 effects on dynamics and GNC, 116–117
 thrust magnitude accuracy, 116
sloshing, 111–113
thermal snap, 122
Internal vibrations, 118–121
International Association of Geomagnetism and Aeronomy (IAGA), 86
International Astronomical Union (IAU), 53–54
International Atomic Time (TAI), 69–70, 1002
International Celestial Reference Frame (ICRF), 53–54
International Earth Rotation and Reference Systems Service (IERS), 55
International Geomagnetic Reference Field (IGRF), 85
International standards, 123

- International Terrestrial Reference Frame (ITRF), 55–56
- Interplanetary reference system, 53–54
- Interpolation, 398–402
coefficients, 428
formulas, 399
inverse, 399–400
methods for tabulated data, 400t
spline, 400–402
tabulated data and finite differences, 399t
- Inverse interpolation, 399–400
- Inverse matrix, 412
- Inverse reinforcement learning, 903–906
feature-matching approaches, 904–906
maximum entropy, 906
- Irregular solar system bodies, 170–174,
803–805
ellipsoidal model, 172–173
mass concentration model, 173
POL, 174
SHE model, 170–171
- Irregular solar system bodies, 132
- Isolation, 634
- J**
- J_2 -perturbed relative dynamics, 189–190
- $J2000$, 55
- Jacchia models, 88–89
- Jacobian matrix, 168
- Jerome Cardan’s Method, 149
- Jitter, 119–120
- Joint Gravity Model-3 (JGM-3), 84
- Joseph form of sequential filters, 536
- Julian dates (JD), 45, 71–73
- K**
- K-means
algorithm, 823, 921
clustering, 825
- Kalman filter(ing) (KF), 448–450,
499–502
approach, 500–501
- Kalman–Bucy filter, 600
- Karl Stumpff’s Method, 149
- Karnopp model, 365
- Keep-out-zone, 785
- Keplerian orbital elements, 389
- Keplerian orbital motion, 78
- Kepler’s First Law of Planetary Motion,
137
- Kepler’s second law of planetary motion,
136
- Keypoints detection network, 879
- Keypoints-based CNN, 877
- KGD STM, 201
- Kinetic energyellipsoid, 231–232
- Knowledge drift error (KDE), 621
- Knowledge reproducibility error (KRE),
621
- Known object, 527
- L**
- Lagrange equation, 103–104
- Lagrange point, 132
- Lagrangian formalism, 106
- Lagrangian function, 104
- Lagrangian points, 163
station-keeping control, 741–742
- Lambert’s problem, 736–737
- Landmarks, 3
- Laplace transform, 1011
- Laser altimeters, 332–333
- Laser scanners, 329
- Laser-based sensors, 332–333
- Latencies, 274
- Latency, 315
- Latitude, 48–49
- Launchand Early Operation (LEOP), 682
- Layer-recurrent neural network, 908
- LCI. *See* Lunar-Centered Inertial (LCI)
- LCROT. *See* Lunar Centered ROTating (LCROT)
- Learning rate, 827–828
- Least squares, 462–463
- Legendre’s polynomials, 81, 171
- Length of Day, 65
- LEO. *See* Low Earth orbit (LEO)
- LEON processor, 923–924
- Levenberg–Marquardt algorithm, 827,
909, 515
- Levenberg–Marquardt curve-fitting
algorithm, 827
- Light detection and ranging (LIDAR),
328–329, 804, 921–922

- processing, 686
strengths and weaknesses, 330t
systems, 254, 504–505
- Light randomization, 888
- Limb detection, 512
- Limb fitting, 512
- Linear discrete-time system, 448
- Linear Fractional Transformation (LFT), 606
- Linear Quadratic Regulator (LQR), 544, 596–600
finite-horizon linear quadratic regulator, 597–598
infinite-horizon linear quadratic regulator, 598
linear quadratic Gaussian control, 598–600
- Linear regression, 826–830, 833
cost function for regression, 826–827
feature engineering and polynomial regression, 828–829
generalization, 829–830
model representation, 826
parameter learning, 827–828
- Linear scale factors, 267–268
- Linear system, 396
- Linear time-invariant systems (LTI systems), 545, 1008, 1011
- Linear-based model, 650
- Linearization, 584–585
of equations of motion, 177–182
process, 180
- Linearized equations of motion, 179
for nearly circular orbits, 183–187
- Linearized relative dynamics, true anomaly parametrization in, 179–182
- Linearized relative motion models, 390
- Load sensitivity function, 558
- Local methods, 508–509
- Local Vertical, Local Horizontal (LVLH), 61, 175, 243, 299–300, 429–431
frame, 403, 571–572
LVLH-fixed attitudes, 943
- Logistic function. *See* Sigmoid function
- Logistic regression, 824–825
for binary classification, 831–834
cost function for binary classification, 833–834
model representation, 831–832
- Long short-term memory (LSTM), 842, 863–864
- Longitude, 48–49
- Loop-shaping design, 563–567
- Loosely coupled architecture, 527–528
- Loss augmentation, 905
- Low Earth orbit (LEO), 14, 79–80, 147, 277, 652, 693, 717, 820
- Low-thrust maneuvers, 731–733
- Low-thrust trajectory design, 737–740
orbit-raising low-thrust trajectory solution, 740f
- Lower triangular matrix, 984
- Luna-3 soviet space probe, 11
- Lunar Centered ROTating (LCROT), 60–61
- Lunar coordinate systems, 59–60
mean earth/polar axis, 59
PA, 59–60
- Lunar-Centered Inertial (LCI), 59
- Lyapunov functions, 584, 588, 743, 913–914
- Lyapunov stability
theorem, 782
theory, 809–810
- Lyapunov theorem, 591–592
- Lyapunov's indirect method, 585
- ## M
- Machine learning (ML), 819, 821–822
- Magnetic dipole moments, 373
- Magnetic field, 85–86, 98–99
- Magnetic moment, 956
- Magnetic potential, 85
- Magnetic torque, 244–245
- Magnetic torquers, 956–960
- Magnetometers, 287–290, 321–322, 946–947, 964, 969
calibration, 798
strengths and weaknesses, 291t
- Magnetoresistive sensors, 288
- Magnetorquers, 338, 369–375, 966
actuation function, 372–373
assembly, 370–372

- Magnetorquers (*Continued*)
control, 757–758
single-axis magnetic control example, 758f
model, 375, 375f
performance, 373–374
- Mahalanobis distance. *See* Standardized distance
- Maneuver calculation algorithms, 437
- Man–Machine Interface (MMI), 678–679
- Maritime navigation, 10–11
- Markov decision process (MDP), 892
- MasCon. *See* Mass Concentration (MasCon)
- Mass Concentration (MasCon), 173
- Mass concentration model, 173
- Mass-spring mechanical models, 111–112
- Mathematical altimeter model, 254
- Mathematical gimbal set, 308
- Mathematical tools, 510
- MathWorks Automotive Advisory Board (MAAB), 666
- MATLAB/Simulink environment, 375, 660–661, 1017
- Matrix/matrices, 38
algebra, 983–991
eigenvectors, 988–990
frobenius norm, 987
matrix inversion, 986–987
matrix multiplication, 985–986
matrix rank, 987–988
singular value decomposition, 990–991
square matrices, 984
inversion, 986–987
analytical computation, 986–987
multiplication, 985–986
properties, 985–986
rank, 987–988
- Max pooling, 851
- Maximum entropy, 906
- Maximum likelihood estimation, 515
- MBSE paradigm. *See* Model-based system engineering paradigm (MBSE paradigm)
- MCU. *See* Microcontroller unit (MCU)
- MDP. *See* Markov decision process (MDP)
- Mean absolute error (MAE), 529
- Mean anomaly, 137
- Mean earth/polar axis, 59
- Mean Earth/Polar Axis reference system (ME reference system), 59
- Mean Earth/Rotation Axis system (MER system), 59
- Mean equator of Date frame (MOD frame), 64
- Mean knowledge error (MKE), 620
- Mean performance error (MPE), 621
- Mean sea level (MSL), 52
- Mean squared error (MSE), 826–827
- Measurement bias, 267
- Measurement errors, 259
- Measurement models, 537
- Measurement noise, 563
rejection, 550
- Mechanical gyroscopes, 313
- Medium Earth orbits (MEO), 147–148, 278, 280
- Medium-Angle Camera (MAC), 325–327
- MEKF. *See* Multiplicative EKF (MEKF); Multiplicative extended Kalman filter (MEKF)
- Meta-reinforcement learning (meta-RL), 920–922
- Metrology, elements of, 257–260
- Microcontroller unit (MCU), 687–688
- Microelectromechanical systems (MEMS), 289, 941–942
- MIMO. *See* Multi-input-multi-output (MIMO)
- Mini-batch learning, 842
- Minimum Impulse Bit (MIB), 350
- Misalignment errors, 273–274, 315
- MISRA. *See* Motor Industry Software Reliability Association (MISRA)
- Mission and Spacecraft Management system (MSM system), 17
- Mission and Vehicle Management (MVM), 17
- Mission and vehicle manager, 643–644
- Mission phases, 25–26

- Mission requirements, 36–37
MKE. *See* Mean knowledge error (MKE)
ML. *See* Machine learning (ML)
MMI. *See* Man–Machine Interface (MMI)
MOD frame. *See* Mean equator of Date frame (MOD frame)
Modal Coordinates, 108
Mode manager, 33–34
Model capacity, 830
Model Predictive Control (MPC), 544, 609–616, 783–787
Model Reference Adaptive Control (MRAC), 601–603
Model-based methods for implementing FDIR systems in GNC applications, 929–931
Model-based reinforcement learning, 902–903
Model-based system engineering paradigm (MBSE paradigm), 927–928
Model-in-the-loop (MIL test), 647, 656–667
modeling of AOCS/GNC algorithms, 656–663
verification activities at, 663–667
algorithms verification and validation, 664
model coverage verification, 666–667
modeling standards verification, 665–666
models profiling, 665
requirements verification, 664–665
Modeling, 1007
Modern spacecraft GNC, 3–10
Modified equinoctial elements, 390
Modified Julian Date (MJD), 72
Modified Rodrigues parameters, 388
Moments of inertia, 228
Momentum capacity, 953
Monte Carlo methods (MC methods), 895–896
analysis, 653
simulations, 650
Moore–Penrose generalized inverse, 361–362
- Motion, integrals of, 134–136
Motor Industry Software Reliability Association (MISRA), 666
MPC. *See* Model Predictive Control (MPC)
MPE. *See* Mean performance error (MPE)
MRAC. *See* Model Reference Adaptive Control (MRAC)
MSE. *See* Mean squared error (MSE)
MSL. *See* Mean sea level (MSL)
MSM system. *See* Mission and Spacecraft Management system (MSM system)
Mu-control, 607–608
Multi-input-multi-output (MIMO), 1007–1008
Multibody spacecraft dynamics, 250–251
Multiconstellation GNSS receivers, 281–282
GNSS strengths and weaknesses, 282t
Multilayer perceptron (MLP), 824–825, 835
Multiple impulse transfer, 418–419
Multiple shooting method, 396
Multiplicative EKF (MEKF), 499–500
Multiplicative extended Kalman filter (MEKF), 875–876
MVM. *See* Mission and Vehicle Management (MVM)
- ## N
- Narrow-Angle Cameras (NAC), 325–327
Navigation, 10, 16, 441–443
absolute attitude navigation, 488–504
absolute orbit navigation, 470–488
AI and, 867–883
AI-based relative pose estimation, 869–874
interface with navigation and uncertainty estimate, 874–877
keypoints regression architecture, 878–883
pose estimation, 867–869
batch estimation, 461–470
budgets, 528–533
convergence, 531–532
effect on GNC chain, 532–533

- Navigation (*Continued*)
 estimation error and filter robustness, 529–531
 camera design, 327–328
 constellations, 147
 filters tuning, 538–540
 grade, 315
 image processing techniques, 507–528
 implementation best practices, 533–540
 design, implementation, tuning, and useful checks for sequential filters, 534–536
 navigation filters tuning, 538–540
 right batch filter, 537–538
 right sequential filter, 533–534
 measurement model, 481–482
 on-board *vs.* ground-based navigation, 443–445
 relative navigation, 504–507
 sequential filters, 445–461
 Net force, 247
 Neural dynamics reconstruction through neural networks, 906–917
 dynamics acceleration reconstruction, 909–915
 fully neural dynamics learning, 907–909
 parametric dynamics reconstruction, 915–917
 Newton’s laws, 131
 Newton’s third law of motion, 342, 354–355
 Nichols plot, 773
 Noise
 amplitude, 271
 density, 271
 errors, 268–272, 313–314
 measurement rejection, 563
 modeling, 270–271
 sensitivity function, 558
 Nondimensional equations of dynamics, 165
 Nondrifting transfer, 417
 Nonlinear autoregressive exogenous model (NARX model), 861, 908
 Nonlinear least square algorithm, 520–521
 Nonlinear programming algorithm (NLP algorithm), 739
 Nonlinear scale factors, 267–268
 Nonlinear systems
 attitude regulation example, 589–592
 control design for, 584–592
 feedback linearization, 586–587
 gain scheduling, 586
 linearization, 584–585
 stability analysis for nonlinear systems, 588–589
 Nonlinearities, presence of, 467
 Nonminimum phase systems, 582–584
 Nonminimum phase transfer function, 582–583
 Nonnormalized quaternion, 424
 Nonorthogonality errors, 273–274
 Normalized Stokes coefficients, 82
 North, East, Down (NED), 58
 Notation rules, 38–42
 Notation table, 39
 NRLMSISE-00 model, 99
 Nutation, 64
 Nyquist criterion, 553
 Nyquist curve, 553–554
- O**
- Object detection network, 856–859
 Objective function, 826–827
 Observation errors, effect of, 464–465
 Observation model, 533–534
 Observer gain matrix, 600
 Ocean tides, 84
 Oil jogs, 358–359
 On-board autonomy, 633–634
 On-board computers (OBCs), 635, 651, 686
 On-board control system, 32–35
 On-board guidance, 382–385
 On-board implementation process
 on-board implementation alternatives, 700–705
 ASIC, 704–705
 FPGA, 702–703
 multiple processors, 701
 processor and multiple DSPs, 701–702
 system-on-chip, 703–704

- on-board processing avionics, 694–700
spacecraft avionics, 686–694
and verification, 705–711
- On-board navigation, 443–445
- On-board orbit sensors, 276–277
- On-board processing avionics, 694–700
accommodation of GNC functions into
on-board SW, 697–700
- On-board processors, 820
- On-board sensor processing, 716,
791–802
autonomous on-board sensor calibration,
796–800, 800f
GNSS-INS integration for on-board
orbit determination, 800–802
sensor failure detection, isolation, and
recovery, 794–796
- On-board software (OBSW), 635, 656,
687–688, 935
accommodation of GNC functions into,
697–700
processors identification for OBC and
GNC functions, 697f
system-level architectures, 698f
models, 1018
- On-ground orbit sensors, 276–277
- One-axis pointing, 423–426, 745–749
maximize secondary target, 749
- One-way radio signals, 284
- Online learning, 842–843
- Open-loop transfer function, 550,
552–554, 564
- Operational sensor models, 255
- Optical data, 284
- Optimization methods, 392, 395
- Orbit sensors, 253, 276–285. *See also*
Global navigation satellite system
sensors (GNSS sensors); Attitude
sensors; Electro-optical sensors;
Inertial sensors; Sun sensors
GNSS sensors, 277–283
ground-based orbit determination,
283–285
- Orbital control systems, 716, 730–742
impulsive and low-thrust maneuvers,
731–733
low-thrust trajectory design, 737–740
- orbital maneuvers, 733–737
station-keeping, 740–742
- Orbital dynamics, 9, 207
irregular solar system bodies, 170–174
relative orbital dynamics, 174–204
three-body problem, 161–170
two-body problem, 132–160
- Orbital elements, 136–138
- Orbital maneuvers, 733–737
coplanar maneuvers, 733–735
Lambert’s problem, 736–737
plane change maneuvers, 735–736
- Orbital period, 148–153
- Orbital perturbations, 154–159
analytical approach, 155–159
argument of periapsis, 159
eccentricity, 157
inclination, 157–158
right ascension of ascending node, 158
semimajor axis, 156
true anomaly, 159
numerical approach, 154–155
- Orbital regimes, 280
- Orbits
attitude, 12
determination, 285, 800–801
operative classification of, 146–148
- Ordinary differential equations, 178
- Otsu’s method, 509
- Out-of-plane
maneuver, 419–420
motion, 405
oscillation, 405
state transition matrix, 407
- Output layer, 836
- Output quantization, 315
- Output saturation, 274
- Over-fit, 830
- P**
- Packet Utilization Standard (PUS),
640–641, 928
- Parabolic orbits, 142, 149
- Parallel axis theorem, 228–229
- Parameters estimation, 458–461
state augmentation for, 458–459
bias estimator, 459

- Parameters estimation (*Continued*)
 use of consider states, 459–461
- Parametric dynamics reconstruction, 915–917
- Parametrization methods, 212
- Parasitic forces
 during thrusters firing, 113–117
 and torques due to plume impingement, 121–122
- Partial derivatives
 calculation, 537–538
 equations, 102–103
- Particle filters (PFs), 455–458
- Particles, 455
- Passive control techniques, 724
- Passive safety, 196–198
- Peak torque, 954
- Pendulum, 111–112
- Performance drift error (PDE), 621
- Performance models, 340, 342
- Performance reproducibility error (PRE), 621
- Performance requirements, 432–433
- Performance sensor models, 256
- Pericenter rotation, argument of, 67
- Perifocal coordinate systems, 61
- Periodic transfer, 416–418
- Perturbation sources, 77–78, 125
- Perturbed relative dynamics with relative orbital elements, 198–200
- Phase ambiguity, 472
- Phase crossover frequency, 555–556
- Phase margin, 556, 564
- Pinhole camera model, 515–519
 camera calibration from known scene, 519
- Pink noise, 270
- Pitch axis, 62
- Pixels
 few tenths of, 526
 pixel-based approach, 508
- Plane, 136
- Planetary bodies, 247–248
- Planetary ephemerides, 95
- Planetary landing guidance, 806–814
 formulation, 806–808
 guidance algorithms, 809–811
- convex optimization method, 811
- polynomial method, 809
- potential field method, 809–810
- pseudospectral method, 810
- zero-effort-miss/zero-effort-velocity method, 810
- hazard avoidance, 814
- sensors and navigation, 811–813
- Planetary models, 45, 73. *See also Earth models*
- coordinate transformations, 63–68
- earth and, 46–53
- GNC, 73–75
- time, 69–73
- Plume impingement, parasitic forces and torques due to, 121–122
- PnP solver for state filter initialization, 882
- Poincaré elements, 390
- Point correspondence-based homography estimation, 515
 maximum likelihood estimation, 515
 robust estimation, 515
- Pointing
 control, 749
 error, 619–620
- Polar motion, 64
- Polarity tests, 970
- Pole placement, 570–577
 for first-, second-, and high-order systems, 577–579
- Polhodes, 231–232
- Policy-based methods, 897
- Polyhedral Model (POL), 174
- Polynomial method, 809
- Pooling layers, 851
- Pose estimation, 521, 867–869
- Position-sensitive devices/quadrant detectors, 948
- Potential field method, 809–810
- Powered descent (PD), 806
- PQW, 61, 429–431
 ECI to, 66–67
- PRE. *See* Performance reproducibility error (PRE)
- Precession, 64–65
- Precise orbit determination (POD), 475, 477–478

- Precise point positioning algorithms (PPP algorithms), 475–476
- Prediction horizon, 611
- Preliminary design, 36–37
- Primer vector, 395
- Principal Axes reference system (PA reference system), 60
- Principal component analysis (PCA), 823
- Printed circuit board (PCB), 371–372
- PCB-embedded torquers, 956
- Probability, 261–265
- Probability distribution function (PDF), 261, 650
- Probability number (PN), 28
- Process noise, 446–447
- Processor-in-the-loop (PIL), 647, 671–672, 709–710
- verification activities at, 676
- Project Requirements Document, 18–20
- Projective geometry, 513–515
- homography, 513
- point correspondence-based homography estimation, 515
- Projective reconstruction, 520–521
- Proper Euler angles, 213
- Proper sensor models, 266
- Proportional integral controllers (PI controllers), 356, 502
- Proportional-derivative controller (PD controller), 566–567, 747
- Proportional-Integral-Derivative control (PID control), 544, 593–596
- Propulsion subsystem design, 344
- Proximal policy optimization algorithm (PPO algorithm), 901–902
- Pseudo-Earth Fixed frame (PEF frame), 64
- “Pseudo” Newtonian inertial system, 54
- Pseudoinertial systems, 55
- Pseudoinverse method, 362
- Pseudorange, 471
- measurements, 279
- Pseudospectral method (PSM), 395–396, 810
- Pulsar-based spacecraft navigation, 480–483
- clock errors, 482–483
- ephemerides error, 483
- Pulse-width modulation (PWM), 944
- PUS. *See* Packet Utilization Standard (PUS)
- Q**
- Q-learning, 897–899
- Q-network, 898
- Quadratic Volterra model (QV model), 200
- Quantization errors, 272–273
- Quasi-Zenith Satellite System (QZSS), 278–279
- Quasisingular relative orbital elements, 191–192
- QUaternion EStimation (QUEST), 495–496
- Quaternions, 212, 217–221, 388, 423–424
- algebra, 994–999
- from two directions, 998–999
- angular velocity, 221–227
- kinematics, 226–227
- logarithm, 429
- relative, 220–221
- rotation, 429–431
- successive rotations, 219–220
- R**
- R2BP. *See* Restricted Two-Body Problem (R2BP)
- Rad-hard (RH), 692
- Radar altimeters, 332–333
- Radial basis function neural network aided adaptive extended Kalman filter (RBFNN-AEKF), 909–910
- Radial basis function neural networks (RBFNNs), 843–845
- Radial basis functions (RBFs), 843
- Radial distance, 55
- Radial impulses, 410
- Random access memory (RAM), 687–688
- Random errors, 261, 268–272
- with uniform distribution, 272
- Random variables, 261–263
- Gaussian random variables, 263
- uniform random variables, 262

- Random walk
 errors, 313–314
 noise, 951
- Range, 283
- RANSAC algorithm, 515
- Rate random walk errors (RRW errors), 314
- Reaction wheel jitter, 119–121
- Reaction wheels, 354–365, 953–955, 965–966, 970
 assembly, 355–357
 control with, 753–756
 desaturation, 716, 755–756
 friction and microvibrations, 357–360
 model, 364–365, 364f
 multiple reaction wheels actuation
 function, 360–363
 performance, 363–364
 strengths and weaknesses, 364t
- Real-time navigation, 475, 478–480
 GNSS-INS integration, 479–480
 Relative GNSS navigation, 480
- Real-time operating system (RTOS), 659–660
- Real-world models (RW models), 1018
- Recovery, 634
- Rectangular coordinate system, 53
- Rectified linear unit activation function
 (ReLU activation function), 837, 908
- Recurrent neural networks (RNNs), 824–825, 859–864
 HNN, 862–863
 LSTM, 863–864
 NARX model, 861
- Red noise, 270
- Reduced-order linear lateral sloshing
 models, 111–112
- Reference systems, 73
 coordinate reference systems, 53–63
 coordinate transformations, 63–68
 earth and planetary models, 46–53
 GNC, 73–75
 time, 69–73
- Reflectivity coefficient (CR), 91
- Region of interest (ROI), 872
- Region proposal-based framework, 856
- Region-based shape representation, 510–511
- moments, 511
- scalar region descriptors, 511
- Regression/classification-based
 framework, 856
- Reinforcement learning, 823–824, 890–906, 920
 deep reinforcement learning algorithms, 897–903
 inverse reinforcement learning, 903–906
 model-based *vs.* model-free, 893t
- Relative attitude dynamics, 249–250
- Relative attitude kinematics, 249
- Relative dynamics modeling
 comparison of, 200–204
 Cartesian and relative orbital elements
 mapping, 203–204
 circular sun-synchronous LEO with
 perigee altitude, 202t
 fixed relative motion geometry with
 small separation, 201t
 using relative orbital elements, 190–200
 coordinates transformation, 193–195
 energy-matching condition and passive
 safety, 196–198
 perturbed relative dynamics with
 relative orbital elements, 198–200
 relative motion geometry, 195–196
- Relative GNC, 775–791
 guidance and control strategies, 779–789
 guidance for relative and proximity
 maneuvers, 775–776
 rendezvous in cislunar space, 789–791
 trajectory design and sensors selection, 776–779
- Relative GNSS navigation, 475–476
- Relative knowledge error (RKE), 621
- Relative motion
 of camera, 524–525
 geometry, 195–196
 for rendezvous guidance applications, 402–409
- Relative navigation, 504–507
 scenarios, 505t
- Relative orbital dynamics, 132, 174–204

- linearization of equations of motion, 177–182
true anomaly parametrization in linearized relative dynamics, 179–182
- linearized equations of motion for nearly circular orbits, 183–187
analysis and characteristic of unperturbed motion, 184–187
- Relative orbital elements (ROEs), 778
mapping, 203–204
perturbed relative dynamics with, 198–200
relative dynamics modeling using, 190–200
- Relative performance error (RPE), 621
- Relative quaternion, 220–221
- Relative representation, 506–507
- Relative wind velocity (*v_{rel}*), 87
- Rendezvous and docking (RVD), 504
- Rendezvous guidance, 402–423
forced motion, 420–423
impulsive maneuvers and trajectories, 410–420
cotangential transfer, 412–414
drift modulation transfer, 418
multiple impulse transfer, 418–419
out-of-plane maneuver, 419–420
periodic transfer, 416–418
trajectory-crossing maneuver, 414–416
two-point transfer, 411–412
- relative motion for rendezvous guidance applications, 402–409
effect of velocity impulses, 409–410
- Rendezvous in cislunar space, 789–791
- Reorientation guidance, 428–429
- Requirements verification, 664–665
code requirement verification, 664–665
models requirement verification, 664
- Residual generation, 930–931
- Resolved object, 527
known object, 527
unknown object, 527
- Restricted Three-Body Problem (RTBP), 162
periodic motion in, 166–170
- Restricted Two-Body Problem (R2BP), 134, 137
- RetinaNet, 856
- Retrofire attitude, 12
- RH. *See* Rad-hard (RH)
- Right ascension, 55
of ascending node, 158
- Right batch filter, 537–538
implementation workflow, 538
- Right sequential filter, 533–534
- Rigid body dynamics, 230–235
angular momentum, 230–231
Euler equation, 232–235
rotational kinetic energy, 231–232
- Ring laser gyros (RLGs), 941–942
- Ritz–Galerkin approximation, 102–103
- RKE. *See* Relative knowledge error (RKE)
- RNNs. *See* Recurrent neural networks (RNNs)
- Robust attitude control of spacecraft, 759–773
robust attitude control synthesis, 768–773
- substructuring modeling, 761–768
assembling of whole spacecraft, 768
flexible solar array with revolute joint, 764–768
main body, 763–764, 764f
- Robust control, 605–609
H-infinity, 605–607
Mu-control, 607–608
robust adaptive controllers, 608–609
robust model predictive control, 616
- Robust estimation, 515
- Robustness margin, 625
- Robustness to uncertainty, 551
- Rodrigues' formula, 81
- Rodrigues parameters, 388–389
- ROEs. *See* Relative orbital elements (ROEs)
- ROI. *See* Region of interest (ROI)
- Roll axis, 62
- Roll-pitch-yaw axes, 62
- Root-mean-square error (RMSE), 477
- Rotation matrix, 274
- Rotation quaternion, 422

- Rotation sequence, 63–64
 Rotational kinetic energy, 231–232
RPE. *See* Relative performance error (RPE)
RRW errors. *See* Rate random walk errors (RRW errors)
RSW, 61–62, 429–431
 ECI to, 67–68
 reference frame, 68
RTBP. *See* Restricted Three-Body Problem (RTBP)
RTOS. *See* Real-time operating system (RTOS)
 Rule-of-thumb, 98
RVD. *See* Rendezvous and docking (RVD)
RW models. *See* Real-world models (RW models)
- S**
- S/C.* *See* Spacecraft (S/C)
SADMs. *See* Solar arrays drive mechanisms (SADMs)
 Sample depletion, 456–457
 Sampling time, 611
 Satellite-based coordinate systems, 61–62
 perifocal coordinate systems, PQW, 61
 satellite body coordinate systems, $b_1 b_2 b_3$, 62–63
 auxiliary satellite body coordinate systems, 62–63
 satellite coordinate system, RSW, 61–62
 Satellites, 71, 241, 280
 altimetry, 330–331
 angular rate, 653
 body coordinate systems, 62–63
 body reference frame, 62
 coordinate system, 61–62
 Saturation, 363
SAVOIR. *See* Space Avionics Open Interface Architecture (SAVOIR)
 Scalar region descriptors, 511
 Scale factors, 267–268
 errors, 267–268, 314
 instability, 314
 models, 268
 ratios, 314
 Scale-invariant feature transform (SIFT), 812–813
 Scanning sensors, 298
 Schmidt coefficients, 85
 Schmidt–Kalman filter, 459–461
 formulations, 461
Schweighart–Sedwick model, 201
SCOE. *See* Special Check-Out Equipment (SCOE)
 Second Cosmic Velocity, 146
 Sectorial Harmonics, 82
 Segmentation, 507
 Selenocentric coordinates, 59
 Semimajor axis, 156
 Sensitivity
 crossover frequency, 561
 functions, 557–563, 581–582
 disturbance rejection, 561–563
 noise measurement rejection, 563
 response to setpoint, 559–561
 Sensor(s), 16, 433–434, 649
 acquisition chain, 266–267
 altimeters, 330–334
 attitude sensors, 286–306
 calibration, 265–266
 electro-optical sensors, 322–329
 faults, 275–276, 795
 inertial sensors, 306–322
 modeling for GNC, 253–274
 models, 255–257, 267, 321–322,
 340–341
 interfaces, 256
 orbit sensors, 276–285
 selection and actuators sizing, 726–730
 magnetorquer sizing, 728t
 Separation principle, 599
 Sequential filtering, 443
 Sequential filters, 445–461
 design, implementation, tuning, and useful checks for, 534–536
 general design considerations, 534–535
 implementation efficiency, 536
 implementation workflow, 535–536
 parameters estimation, 458–461
 for spacecraft navigation, 447–458
 working principle, 446–447

- Sequential measurements, 536
Serial Peripheral Interface (SPI), 258
Seven-point algorithm, 524
Severity number (SN), 28
Sextants, 3
SEZ. *See* South, East, Zenith (SEZ)
SGD. *See* Stochastic gradient descent (SGD)
Shallow neural network, 836
SHE model. *See* Spherical Harmonics Expansion model (SHE model)
Shooting method, 396
Shuster, 495
Sidereal time, 64, 69
SIFT. *See* Scale-invariant feature transform (SIFT)
Sigma points, 452
Sigmoid function, 831, 837
Signal-to-noise ratio (SNR), 324, 867–868
Simple spin, 234
Simplex algorithm, 350
Simulink Configuration Parameters, 1023–1026
Simulink GNC models, 656
Simultaneous Localization and Mapping (SLAM), 527
Single instruction, multiple data (SIMD), 692
Single Shot MultiBox Detector (SSD), 856
Single-event upsets (SEUs), 692
Single-input single-output channel (SISO channel), 760, 1007–1008
Singular Value Decomposition (SVD), 492, 987, 990–991
Skew-symmetric matrix, 984
Sliding Mode Control (SMC), 544, 616–618
Slosh motion, 112–113
Sloshing, 111–113
Small bodies, 170
Small satellites/CubeSats, 938–971
hardware limitations, 940–945
burden of miniaturization, 941
pointing performances, 943
size and mass limitation, 941–943
thrusters, 943–945
SMP. *See* Symmetric multiprocessing (SMP)
SN. *See* Severity number (SN)
SoC. *See* System-on-chip (SoC)
Softmax activation function, 837
Software (SW), 632, 685, 1017
algorithm, 718
design process, 36
development process, 36
rendering, 888–890
Software Requirements Document, 36–37
Software Verification and Validation plan, 36–37
Software-in-the-loop (SIL), 647, 671, 968–969
SIL/PIL test, 667–676
autocoding, 668–671
processor-in-the-loop, 671–672
software-in-the-loop, 671
verification activities at SIL level, 672–676
verification activities at, 672–676
code coverage verification, 675–676
code functional verification, 673–674
code standards verification, 674–675
requirements verification, 674
SOI. *See* Sphere of influence (SOI)
Solar arrays, 104
Solar arrays drive mechanisms (SADMs), 718–719
Solar flux, 90
Solar panels pointing, 759
Solar radiation, 93–94, 99–100
Solar radiation pressure (SRP), 77, 89–94, 246
Albedo and infrared emission, 93–94
eclipse, 92–93
torque, 246–248
Solar System Barycenter (SSB), 481
Solar time, 69
Solution method, 397
Solver, 1024, 1024f
South, East, Zenith (SEZ), 58
Space
AI in, 821–866

- Space (*Continued*)
 craftsystem engineers, 4–5
 industry, 631
 mission, 382–383
 segment, 285, 382–383
 system, 34
- Space Avionics Open Interface
 Architecture (SAVOIR), 700
- Space environment, 124
 external perturbations, 79–97
 modeling guidelines, 97–100
 GNC, 124–126
 internal perturbations, 100–122
 modeling guidelines, 123–124
 models, 126
 perturbation sources, 78
- Spacecraft (S/C), 925–926
 attitude, 209
 dynamics, 230
 avionics, 686–694
 application-specific integrated circuit, 693
 digital signal processor, 688–689
 field programmable gate array, 689–691
 general-purpose processor or microcontroller, 687–688
 graphical processing unit, 689
 system-on-chip, 694
- control, 546–547
 navigation, 443, 527–528
 design process, 528
 EKF, 450–452
 $H\infty$ filter, 450
 KF, 448–450
 PF, 455–458
 sequential filters for, 447–458
 UKF, 452–454
- Spacecraft PosE Estimation Dataset (SPEED), 870
- Spacecrafts, 113–114
 sensors, 265
- Spaceflight, 3
- Special Check-Out Equipment (SCOE), 679–680
- Special orthogonal matrices, 210
- Specific angular momentum, 134
- Specific energy, 135–136
- Specific orbital energy, 135–136
- Speed estimation, 356
- Sphere of influence (SOI), 160
- Spherical Harmonics Expansion model (SHE model), 81, 170–171
- SPI. *See* Serial Peripheral Interface (SPI)
- Spike fault, 275
- Spline interpolation, 400–402
- Square matrices, 984
- Square Root Information Filter (SRIF), 469–470
- SSD. *See* Single Shot MultiBox Detector (SSD)
- SSOs. *See* Sun-synchronous orbits (SSOs)
- Stability, 552–557
 analysis, 563
 of dual spin attitude dynamics, 240
 for nonlinear systems, 588–589
 in state space, 569
- equation, 238
 margins, 552–557
 time, 620
- Standard deviation, 1000
- Standard gravitational parameter, 134
- Standardized distance, 1002
- Stars, 302–303
 sensors, 300–306
 strengths and weaknesses, 306
- trackers, 768, 949–950, 964, 969
- tracking, 302
- State estimator, 882–883
- State feedback control law, 570
- State filter initialization, PnP solver for, 882
- State prediction, 448
- State Transition Matrix (STM), 167–168, 404, 407–408, 463
- State variables, 1007–1008
- State vector, 507
- State-space control law, 580
- State-space equations, 1008
- State-space models, 1007–1009
- State-space vector, 1009
- State-transition matrix, 1008
- Static and dynamic performance, 549–550

- Static attitude determination, 489
Static imbalance, 359
Static sensors, 298
Station-keeping control, 740–742
Stationary coplanar elliptical relative orbit, 187
Statistical error representation, 317–321
Statistics, 1000
 basics of, 1000–1002
 scalar statistics parameters, 1000
 vector and matrix forms of statistic quantities, 1001–1002
Steady-state error, 550
Stereo camera, 776–777
Stiction, 357
Stochastic gradient descent (SGD), 842
Stokes coefficients, 81, 171
Strapdown accelerometers, 308
Stribeck effect, 358
Structured singular value, 625
Stuck fault, 276
Sub-Earth point, 59
Subpixel, 526
Sun presence sensors, 296
Sun sensors, 291–297, 947–949, 963–964, 970. *See also* Orbit sensors; Attitude sensors; Electro-optical sensors; Global navigation satellite system sensors (GNSS sensors); Inertial sensors
analog sun sensors, 292–294
digital, 295–296
model, 296–297
strengths and weaknesses, 297t
sun presence sensors, 296
 types of, 296t
Sun-synchronous orbits (SSOs), 148, 943
Sun–Earth SOI borders, 161
Sun–Earth system, 248
Supervised learning, 826–843
 algorithm, 822–823
 artificial neural networks for multiclass classification, 834–843
 linear regression, 826–830
 logistic regression for binary classification, 831–834
SVD. *See* Singular Value Decomposition (SVD)
Symmetric matrix, 984
Symmetric multiprocessing (SMP), 696–697
System definition, 27
System modeling, 433–434
System redundancy, 24–25
System-on-chip (SoC), 685, 694, 703–704
 DSP or GPU SoCs, 704
 FPGA, 704
- T**
- Tactical grade, 315
Tailoring process, 20
Tait–Bryan angles, 213
Tangential impulses, 410
Target network, 898–899
Technology readiness level 9 (TRL 9), 945
Telecommands (TCs), 640–641, 718–719
Telemetry (TM), 636
Telemetry/telecommand transmitters (TM/TC transmitters), 658, 678–679
Temperature sensitivity, 951
Temperature-dependent bias, 313
Temporal difference method (TD method), 895–896
Temporal discretization, 274
Terrestrial Time (TTT), 1004
Test error, 829
Testbed for Rendezvous and Optical Navigation (TRON), 883–884
Texture randomization, 888
Thermal Infrared cameras (TIR cameras), 325–327
Thermal snap, 122
Thermochemical processes, 342
Third Cosmic Velocity, 146
Third-body perturbation, 94–95
 modeling guidelines, 100
Three-axis pointing, 750–753
 two loops, 752–753

- Three-body problem (3BP), 132, 161–170. *See also* Two-body problem
 attitude dynamics, 248–249
 circular restricted three-body problem, 162–164
 elliptic restricted three-body problem, 164–166
 environments, 248
 periodic motion in restricted three-body problem, 166–170
 circular restricted three-body problem, 167–169
 elliptic restricted three-body problem, 169–170
- Three-body synodic coordinate systems, 60–61
- Three-dimension (3D), 280–281
 point computation, 525
 space, 280–281
 vision, 512–521
 multiple views scene reconstruction, 520–521
 pinhole camera model, 515–519
 pose estimation, 521
 projective geometry, 513–515
 projective reconstruction, 520–521
 triangulation, 520
- Three-element error parametrization, 875–876
- Three-way radio signals, 284
- Threshold-based diagnostic routines
 process symptoms, 642
- Throttleability, 944
- Thrust magnitude accuracy, 116
- Thrust Management Function (TMF), 347–353
 example thruster configuration data, 349t
 schematic of closed-loop system, 347f
 thrusters strengths and weaknesses, 353t
- Thrusters, 344–354, 729–730, 943–945
 assembly, 345–346
 firing, parasitic forces and torques during, 113–117
 model, 353–354, 353f
 plumes, 302–303
- thrust management and actuation function, 347–353
- Tightly coupled architecture, 527–528
- Time, 45, 69–73
 Julian dates, 71–73
 universal time, 70–71
- Time accuracy, 73–74
- Time delays, 583
- Time intervals, 73
- Time laws, 148–153
- Time of Arrival (TOA), 481
- Time of flight (TOF), 732
- Topocentric coordinate systems, 57–59
- Topocentric declination, 57
- Topocentric equatorial coordinate systems, 57
- Topocentric horizon coordinate systems, 57–59
- Topocentric reference spherical coordinates, 59
- Topocentric right ascension, 57
- Torques, 230
 noise, 359
 during thrusters firing, 113–117
- Total Electron Content (TEC), 473
- Total ionizing dose (TID), 694–695
- Training error, 829
- Training set, 829–830
- Trajectory design, 776–779
- Trajectory optimization methods, 395–396
- Trajectory-crossing maneuver, 414–416
- Transduction, 258
- Transfer functions, 1011–1015
 of closed-loop system, 557–558
 interconnection of linear systems, 1014f
- Transfer learning (TL), 920–921
- Transfer polynomials, 416
- Transformation matrices, 63
- Triad, 470–488
- Triangulation, 520
- True anomaly, 151, 159
 parametrization in linearized relative dynamics, 179–182
- True of Date frame (TOD frame), 64
- Tsiolkovsky's rocket equation, 344
- Tuning for sequential filters, 534–536

- Two degrees of freedom control, 567
Two-axis gyros, 310–311
Two-axis pointing, 426–427
 extended vector normalization, 427
 quaternion rotation, 429–431
 reorientation, 428–429
Two-body problem, 132–160. *See also*
 Three-body problem (3BP)
 energetic analysis and cosmic velocities,
 143–146
 geometrical classification of conics,
 140–143
 integrals of motion and orbital elements,
 134–140
 integrals of motion, 134–136
 orbital elements, 136–138
 TLE, 139–140
 operative classification of orbits, 146–148
 geosynchronous/geostationary orbits,
 147
 low earth orbits, 147
 MEO, 147–148
 SSO, 148
orbital perturbations, 154–159
time laws and orbital period,
 148–153
 circular orbits, 148–149
 elliptic orbits, 150–151
 hyperbolic orbits, 152
 parabolic orbits, 149
 time law solution methods, 154t
 universal time law, 152–153
validity range of, 160
Two-body problem, 131
Two-dimension (2D), 828–829
 input, 828–829
 schematization of array, 106
shape representation, 509–512
 applicative case, 511–512
 chain codes, 510
 contour-based shape representation,
 510
 fourier transforms, 510
 geometric boundary-based features,
 510
 region-based shape representation,
 510–511
- Two-input two-output Port (TITOP),
 761
Two-Line Elements (TLEs), 139–140
 characteristic parameters of conics, 144t
Two-point transfer, 411–412
Two-views geometry, 521–525
 epipolar geometry, 521–523
 relative motion of camera, 524–525
 camera matrix and 3D point
 computation, 525
 fundamental matrix estimation, 524
 stereo correspondence, 525
Two-way radio signals, 284
- U**
- U-D filter, 470
Underfit, 830
Uniform distribution, random errors with,
 272
Uniform random variables, 262
Unit quaternions, 388
Universal [Synchronous and]
 Asynchronous Receiver-
 Transmitter (U[S]ART), 258
Universal approximation theorem,
 834–835, 906, 912–913
Universal Time (UT1), 65
Universal time, 70–71
 law, 152–153
Unknown object, 527
Unperturbed motion
 analysis and characteristic of, 184–187
 circular relative orbit, 186–187
 comparison of relative dynamics
 modeling, 200–204
 concentric coplanar absolute orbit,
 185–186
 impulsive shots, 187
 impulsive motion in linearized relative
 dynamics for nearly circular orbits,
 189t
 J2-perturbed relative dynamics, 189–190
 relative dynamics modeling using relative
 orbital elements, 190–200
 stationary coplanar elliptical relative orbit,
 187
Unscented Kalman filter (UKF), 452–454

Unsupervised learning, 823, 825
 Upper triangular matrix, 984
 US Standard Atmosphere, 88
 User Requirements Document, 36–37
 UTC. *See* Coordinated Universal Time (UTC)

V

Vacuum core, 956
 Validation
 of AI-based systems, 883–890
 CNN validation, 884–888
 training augmentation techniques, 888–890
 validation of keypoints detection accuracy, 890
 process, 648
 of ROE formulation, 203

Value-based methods, 897
 Vanishing gradients, 841–842
 Variance, 318

Vectors, 38
 identities, 991–994
 cross product, 992–994
 dot product, 992
 outer product, 994
 vector norm, 992

Velocity impulses
 effect of, 409–410
 comparison of tangential *vs.* radial impulses as means to affect trajectory changes, 411t
 Velocity random walk errors (VRW errors), 314
 Venus express monitoring camera (VMC), 692
 Verification and validation process (V&V process), 663, 929
 Verification process, 648

Very Long Baseline Interferometry— Δ -DOR (VLBI— Δ -DOR), 284
 Very long instruction word (VLIW), 700
 Video processor units (VPUs), 820
 microprocessor, 923–924
 Viscous friction, 357
 Visible/infrared cameras, 948

W

Wahba problem, 491–496
 davenport q-method, 493–495
 QUEST method, 495–496
 SVD method, 492
 Waterfall plot, 118–119
 Weights update rule, 915
 WGS-84 model, 46–47
 White noise, 270
 Wide-Angle Cameras (WAC), 325–327
 Worst-case analysis (WCA), 655

X

Xilinx, 925

Y

Yamanaka–Ankersen STM, 201
 Yan–Alfriend nonlinear theory, 201
 Yaw axis, 62
 Yaw-pitch-roll angles, 213
 You Only Look Once (YOLO), 856

Z

Zero-crossing, 357, 955
 Zero-effort-miss/zero-effort-velocity method (ZEM/ZEV method), 810
 Zero-elevation angle, 350
 Zonal harmonics, 82

Modern Spacecraft Guidance, Navigation, and Control

From System Modeling to AI and Innovative Applications

Edited by Vincenzo Pesce, Andrea Colagrossi, and Stefano Silvestrini

Modern Spacecraft Guidance, Navigation, and Control: From System Modeling to AI and Innovative Applications provides a comprehensive foundation of theory and applications of spacecraft GNC, from fundamentals to advanced concepts, including modern AI-based architectures with focus on hardware and software practical applications. Divided into four parts, this book begins with an introduction to spacecraft GNC, before discussing the basic tools for GNC applications. These include an overview of the main reference systems and planetary models, a description of the space environment, an introduction to orbital and attitude dynamics, and a survey on spacecraft sensors and actuators, with details of their modeling principles. Part 2 covers guidance, navigation, and control, including both on-board and ground-based methods. It also discusses classical and novel control techniques, failure detection isolation and recovery (FDIR) methodologies, GNC verification, validation, and on-board implementation. The final part 3 discusses AI and modern applications featuring different applicative scenarios, with particular attention on artificial intelligence and the possible benefits when applied to spacecraft GNC. In this part, GNC for small satellites and CubeSats is also discussed.

Modern Spacecraft Guidance, Navigation, and Control: From System Modeling to AI and Innovative Applications is a valuable resource for aerospace engineers, GNC/AOCS engineers, avionic developers, and AIV/AIT technicians.

Key Features

- Provides an overview of classical and modern GNC techniques, covering practical system modeling aspects and applicative cases.
- Presents the most important artificial intelligence algorithms applied to present and future spacecraft GNC.
- Describes classical and advanced techniques for GNC hardware and software verification and validation , as well as GNC failure detection isolation and recovery (FDIR).

About the Editors

Dr. Vincenzo Pesce is a GNC Engineer at Airbus D&S Advanced Studies department in Toulouse. His current research interests include autonomous GNC for proximity operations, rendezvous and landing, vision-based navigation, and GNC innovative methods.

Dr. Andrea Colagrossi is an Assistant Professor of Flight Mechanics at the Aerospace Science and Technology Department of Politecnico di Milano. His main research interests are spacecraft GNC and system engineering for advanced small satellite applications, with a focus on effective GNC implementation with limited hardware resources, innovative GNC techniques, and autonomous failure and contingency modes management.

Dr. Stefano Silvestrini is a Postdoctoral Researcher at the Aerospace Science and Technology Department of Politecnico di Milano. His research interests include the development of artificial intelligence algorithms for autonomous GNC in distributed space systems and proximity operations, particularly tailored for embedded applications in small platforms.



ELSEVIER

elsevier.com/books-and-journals

ISBN 978-0-323-90916-7



9 780323 909167