Optimal Control Problem Solved via Swarm Intelligence

Dario Spiller

February 19, 2018
Version: Rev. 1

Sapienza University of Rome

SAPIENZA
UNIVERSITÀ DI ROMA

Department of Mechanical and Aerospace Engineering

Ph.D. Thesis - Cycle XXX

Optimal Control Problem Solved via Swarm Intelligence

Dario Spiller

1. Reviewer Prof. Robert G. Melton
Department of Aerospace Engineering
The Pennsylvania State University

2. Reviewer Prof. Riccardo Bevilacqua
Department of Mechanical and Aerospace Engineering
University of Florida

Supervisor Prof. Christian Circi

February 19, 2018

Dario Spiller

Optimal Control Problem Solved
via Swarm Intelligence

Ph.D. Thesis - Cycle XXX, February 19, 2018
Reviewers: Prof. Robert G. Melton and Prof. Riccardo Bevilacqua
Supervisor: Prof. Christian Circi

Sapienza University of Rome

Department of Mechanical and Aerospace Engineering
Via Eudossiana 18
Rome 00184, Italy

Abstract

This thesis deals with solving optimal control problems via swarm intelligence. Great emphasis is given to the formulation of the optimal control problem regarding fundamental issues such as unknowns identification, numerical transcription and choice of the nonlinear programming solver. The Particle Swarm Optimization is taken into account, and most of the proposed problems are solved using a differential flatness formulation. When the inverse-dynamics approach is used, the transcribed parameter optimization problem is solved assuming that the unknown trajectories are approximated with B-spline curves. The Inverse-dynamics Particle Swarm Optimization technique, which is employed in the majority of the numerical applications in this work, is a combination of Particle Swarm and differential flatness formulation. This thesis also investigates other opportunities to solve optimal control problems with swarm intelligence, for instance using a direct dynamics approach and imposing a-priori the necessary optimality conditions to the control policy. For all the proposed problems, results are analyzed and compared with other works in the literature. This thesis shows that metaheuristic algorithms can be used to solve optimal control problems, but near-optimal or optimal solutions can be attained depending on the problem formulation.

Abstract (Italiano)

Questa tesi descrive come risolvere problemi di controllo ottimo tramite *swarm intelligence*. Grande enfasi viene posta circa la formulazione del problema di controllo ottimo, in particolare riguardo a punti fondamentali come l'identificazione delle incognite, la trascrizione numerica e la scelta del risolutore per la programmazione non lineare. L'algoritmo Particle Swarm Optimization viene preso in considerazione e la maggior parte dei problemi proposti sono risolti utilizzando una formulazione *differential flatness*. Quando viene usato l'approccio di dinamica inversa, il problema di ottimo relativo ai parametri di trascrizione è risolto assumendo che le traiettorie da identificare siano approssimate con curve B-splines. La tecnica Inverse-dynamics

Particle Swarm Optimization, che viene impiegata nella maggior parte delle applicazioni numeriche di questa tesi, è una combinazione del Particle Swarm e della formulazione differential flatness. La tesi investiga anche altre opportunità di risolvere problemi di controllo ottimo tramite swarm intelligence, per esempio usando un approccio di dinamica diretta e imponendo a priori le condizioni necessarie di ottimalitá alla legge di controllo. Per tutti i problemi proposti, i risultati sono analizzati e confrontati con altri lavori in letteratura. Questa tesi mostra quindi the algoritmi metaeuristici possono essere usati per risolvere problemi di controllo ottimo, ma soluzioni ottime o quasi-ottime possono essere ottenute al variare della formulazione del problema.

Contents

List of Figures

List of Tables

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor Prof. C. Circi for the continuous support of my Ph.D. study and related research. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I truly thank the person who first believed in me and in my research ability. He encouraged me to undertake abroad research activities and inspired most of the work reported in this manuscript. For these reasons, and others that I am not going to write here, I sincerely thank Prof. F. Curti.

I would like to thank my thesis reviewers, Prof. R. G. Melton and Prof. R. Bevilacqua, for their insightful comments, their encouragement and the helpful hints which stimulated me to widen my research from various perspectives. They have been wonderful advisors during my research periods in State College and Gainesville as visiting scholar.

My sincere thanks also goes to Dr. G. Di Mauro and Dr. K. Basu who have supported my research activity and helped me writing and reviewing journal and conference papers. Besides the research activity, they have been, and they are, sincere friends who have spent with me wonderful moments in the USA.

I sincerely thank my colleagues Alessandro, Daniele, Marco and Riccardo of ARCA Dynamics for having helped and assisted me while trying to combine university and company duties. The group has been a source of friendships as well as good advice and collaboration. I thank all the lab-mates from the Arca Laboratory who have collaborated with me through these years. I am particularly grateful to Vincenzo for his continuous and reliable collaboration and to Danilo for the time dedicated to me whenever I needed his help.

I could not have enjoyed these years without the faithful support of my fiancee, who always believed in my activity and supported me during difficult periods. She spent with me most of the time in USA making all those experiences unique and unforgettable. She has been my first love when I was a teenager and she is still

an essential part of me. For her love and her continuous and precious assistance, I thank the love of my life, Federica.

Last but not the least, I would like to thank my family: my parents, my brother and my sister. They have always been a loving, supportive and encouraging reference point through all my life, and they will always be.

Dario Spiller

February 2018

Publications

In this section the list of the journal and conference papers produced within the Ph.D. period is reported. The collaboration with industries (*Finmeccanica Leonardo*) and with other Universities in the U.S. (the *Pennsylvania State University* and the *University of Florida*) has led to the publications of several works, some of them related to the Ph.D. activity and others not. The material contained in the following papers constitutes the main source of information for this work. The reader may refer to them to get further details concerning the theory described in this thesis.

Works related to the Ph.D. activity

Journal papers

- Spiller, D., Ansalone, L., and Curti, F., "Particle Swarm Optimization for Time-Optimal Spacecraft Reorientation with Keep-Out Cones," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 2 (2016), pp. 312-325. doi: 10.2514/1.G001228

- Spiller, D., Curti, F., Circi, C., "Minimum-Time Reconfiguration Maneuvers of Satellite Formations Using Perturbation Forces," *Journal of Guidance, Control and Dynamics*, Vol. 40, No. 5 (2017), pp. 1130-1143. doi: 10.2514/1.G002382

- Spiller, D., Melton, R. G., Curti, F., "Inverse Dynamics Particle Swarm Optimization Applied to Constrained Minimum-Time Maneuvers Using Reaction Wheels," *Aerospace Science and Technology*, accepted (2017). In print. doi: 10.1016/j.ast.2017.12.038

- Spiller, D., Circi, C., Curti, F., "Particle Swarm Optimization with Domain Partition and Control Assignment for Minimum-Time Maneuvers," *Journal of Guidance, Control and Dynamics*, accepted. Publication Date (online): November 16, 2017. In print. doi: 10.2514/1.G002980

- Spiller, D., Curti, F., Ansalone, L., "Inverse-Dynamics Particle Swarm Optimization for Spacecraft Minimum-Time Slew Maneuvers with Constraints," *Aerotecnica Missili & Spazio, The Journal of Aerospace Science, Technology and Systems*, Vol. 96, No. 3 (2017), pp. 111-123. doi: 10.19249/ams.v96i3.302.

- Spiller, D., Basu, K., Curti, F., Circi, F., "On the optimal passive formation reconfiguration by using attitude control," *Acta Astronautica*. Accepted, in print. Publication Date (Online): 6 February 2018. doi: 10.1016/j.actaastro.2018.01.052

- Parente, D., Spiller, D., Curti, F., "Near-Optimal Maneuvers for Satellite Formations with Inverse Dynamics Approach and Differential Evolution," *Journal of Guidance, Control and Dynamics*. Accepted, in print.

Conference papers

- Spiller, D., Curti, F., Ansalone, L., "Inverse Dynamics Particle Swarm Optimization for Spacecraft Minimum-Time Maneuvers with Constraints," *23rd Conference of the Italian Association of Aeronautics and Astronautics*, Politecnico di Torino, Turin, Italy, November 17th - 19th, 2015.

- Spiller, D., Curti, F., "Inverse Dynamics Particle Swarm Optimization for Nanosatellites Rendezvous via Differential Drag," *3rd IAA Conference*, Rome, Italy, November 30th - December 5th, 2015.

- Spiller, D., Curti, F., Melton, R. G., "Inverse Dynamics Particle Swarm Optimization Applied to Constrained Minimum-Time Maneuvers Using Reaction Wheels," *67th International Astronautical Congress*, Guadalajara, Mexico, September 26th - 30th, 2016.

- Spiller, D., Basu, K., "Optimal Passive Formation Reconfiguration using Attitude Control and Perturbing Forces," *9th International Workshop on Satellite Constellations and Formation Flying*, Boulder, Colorado, June 19th - 21th, 2017.

- Spiller, D., Melton, R. G., Curti, F., "Inverse dynamics particle swarm optimization applied to Bolza problems," *Proceedings of the AAS 2017 in Advances in the Astronautical Sciences Guidance, Navigation and Control 2017*, Stevenson, Washington, US, August 21th - 24th, 2017.

Works related to other research activities

Journal papers

- Spiller, D., Stabile, A., Lentini, D., “Design and Testing of a Demonstrator Electric–Pump Feed System for Liquid Propellant Rocket Engines," Aerotecnica Missili & Spazio, The Journal of Aerospace Science, Technology and Systems, Vol. 92, 10/2014, pp. 123-130. doi: 10.19249/ams.v92i3-4.99

- Schiattarella, V., Spiller, D., Curti, F., “A novel star identification technique robust to high presence of false objects: The Multi-Poles Algorithm,” *Advances in Space Research*, Vol. 59, No. 8 (2017), pp. 2133-2147. doi: 10.1016/j.asr.2017.01.034

- Di Mauro, G., Bevilacqua, R., Spiller, D., Sullivan, J., D’Amico, S., “Continuous maneuvers for spacecraft formation flying reconfiguration using relative orbit elements,” *Acta Astronautica*. Accepted, in print.

- Di Mauro, G., Spiller, D., Bevilacqua, R., D’Amico, S., “Spacecraft Formation Flying Reconfiguration with Extended and Impulsive Maneuvers,” *Journal of the Franklin Institute*, submitted and under review.

Conference papers

- Curti, F., Spiller, D., Ansalone, L., Becucci,S., Procopio, D., Boldrini, F., Fidanzati, P., Sechi, G., “High angular rate determination algorithm based on star sensing,” *Proceedings of the AAS 2015 in Advances in the Astronautical Sciences Guidance, Navigation and Control 2015*, Vol. 154, Vail, Colorado, US, August 9th - 13th, 2015.

- Curti, F., Spiller, D., Ansalone, L., Becucci,S., Procopio, D., Boldrini, F., Fidanzati, P., “Determining high rate angular velocity from star tracker measurements,” *66th International Astronautical Congress*, Jerusalem, Israel, October 12th - 16th, 2015.

- Di Mauro, G., Bevilacqua, R., Spiller, D., D’Amico, S., “Continuous maneuvers for spacecraft formation flying reconfiguration using relative orbit elements,” *9th International Workshop on Satellite Constellations and Formation Flying*, Boulder, Colorado, June 19th - 21th, 2017.

- Curti, F., Spiller, D., Ansalone, L., “Recognition of Orbiting-Objects Through Optical Measurements of Light-Reflecting-Targets by Using Star-Sensors,” *Pro-*

ceedings of the 1st IAA Conference on Space Situational Awareness (ICSSA), Orlando, Florida, US, Nov. 13-15, 2017

- Schiattarella, V., Spiller, D.,Curti, F., "Efficient star identification algorithm for nanosatellites in harsh environment,"*4th IAA Conference on University Satellite Missions and CubeSat Workshop*, Rome, Italy, 2017

- Di Mauro, G., Bevilacqua, R., Spiller, D., Curti, F., "Optimal Continuousl Maneuver for Satellite Formation Reconfiguration in J2-Perturbed Orbits," *Proceedings of the 2018 AIAA Space Flight Mechanics Meeting, AIAA Science and Technology Forum and Exposition 2018*, Kissimmee, Florida, US, 8-12 January, 2018

Acronyms

In alphabetical order

ATF	=	Along Track Formation
BC	=	Boundary Constraints
BOCP	=	Bolza Optimal Control Problem
DE	=	Differential Evolution
DF	=	Differential Flatness
DI	=	Differential Inclusion
DD	=	Differential Drag
DPSO	=	Direct-dynamics Particle Swarm Optimization
DSRP	=	Differential Solar Radiation Pressure
EC	=	Evolutionary Computation
FSS	=	Feasible Search Space
GCF	=	General Circular Formation
GEO	=	Geostationary Equatorial Orbit
GG	=	Gravity Gradient
GVE	=	Gauss Variational Equations
HCW	=	Hill-Clohessy-Wiltshire
IPSO	=	Inverse-dynamics Particle Swarm Optimization
KKT	=	Karush–Kuhn–Tucker
LEO	=	Low Earth Orbit
LP	=	Linear Programming
MEO	=	Medium Earth Orbit
MRP	=	Modified Rodrigues Parameter
NLP	=	Non-Linear Programming
OCP	=	Optimal Control Problem

PC	=	Path Constraints
PCF	=	Projected Circular Formation
PMP	=	Pontryagin Minimum Principle
POCS	=	Pseuodospectral Optimal Control Software
POP	=	Parameter Optimization Problem
PSO	=	Particle Swarm Optimization
ROE	=	Relative Orbit Elements
SC	=	State-Control
SRP	=	Solar Radiation Pressure
SS	=	Search Space
TPBVP	=	Two-Point Boundary-Value Problem

Preface

This thesis investigates new numerical approaches to solve optimal control problems. The most important feature of the proposed solutions is the employment of meta-heuristic techniques to cope with the parameter optimization problems. Summarizing the research activity that has already been published in journals and presented at several international conferences, the Particle Swarm optimization is the method that is mainly proposed. Another fundamental feature characterizing most of the chapters of this thesis is the differential flatness formulation of the dynamical systems. The combination of the Particle Swarm and the differential flatness is the core characteristic of the Inverse-dynamics Particle Swarm Optimization technique which is employed to solve most of the problems presented in the thesis. However, some chapters are dedicated to the solution of optimal control problems via other numerical methods, employing direct dynamics approaches or using different technique for the parameters optimization. The structure of the thesis is outlined below.

Part I The fundamental concepts behind the optimal control theory are introduced and the parameters transcription required for numerical solutions is described. In further details, this part is organized as follows.

- *Chapter 1*. The most important concepts concerning optimization, optimal control and planning of maneuvers for space applications are presented. The relationship between optimal control problem and parameter optimization is explained and the reasons behind the choice of an heuristic technique are given.

- *Chapter 2*. The mathematical statement of an optimal control problem is introduced. Different formulations are reported to underline the impact of the problem setting on the transcribed parameters optimization problem. The differential flatness formulation employed in the Inverse-dynamics Particle Swarm Optimization is described.

Part II The common features of the numerical solutions presented in the thesis are outlined. In further details, this part is organized as follows.

- *Chapter 3*. The Particle Swarm Optimization algorithm is presented. The mathematical features of the different paradigms (global, local and unified) are presented. The exploration and exploitation abilities are described, as their meanings is fundamental to set properly the PSO parameters and allow one to find the searched-for solution to the optimal control problem.

- *Chapter 4*. The Inverse-dynamics Particle Swarm Optimization is introduced. The main important features are described, such as the differential flatness implementation, the constraint handling technique and the approximation of the flat output by means of B-spline curves.

Part III This part deals with spacecraft reorientation maneuvers. Different problem formulations are proposed, and a major emphasis is given to the Inverse-dynamics Particle Swarm Optimization. In further details, this part is organized as follows.

- *Chapter 5*. This chapter is based on Refs. [1, 2, 3, 4]. The Inverse Dynamics Particle Swarm Optimization is employed to solve the problem of spacecraft time-optimal reorientation maneuvers. Boundaries and path constraints are considered. It is established that near time-optimal solutions satisfying all the boundary and path constraints can be evaluated.

- *Chapter 6*. This chapter shows that the Inverse-dynamics Particle Swarm Optimization can be used to find feasible near-optimal solutions for difficult problems with nonconvex state constraints and nonconvex cost functions. Minimum-time, minimum-energy and minimum-effort maneuvers are addressed considering the constrained slew-maneuver as a test case.

- *Chapter 7*. In this chapter direct dynamics method is presented. The Particle Swarm Optimization is employed to search for minimum-time maneuvers assigning a bang-bang control policy and dividing the search space into several sub-domains. The Push In and Push Out features are introduced. Two different test cases are reported to validate the method by comparison with other results from the literature.

Part IV This part deals with formation flying reconfiguration maneuvers. In further details, this part is organized as follows.

- *Chapter 8*. Minimum-time reconfiguration of satellite formations are proposed considering the perturbation forces as control variables. The Inverse

Dynamics Particle Swarm Optimization is employed. The evolution of the configuration is simulated with a high-fidelity orbital simulator considering all the perturbations that can affect the maneuver.

- *Chapter 9*. Near time-optimal maneuvers performed by satellite formations during proximity operations and reconfiguration maneuvers are evaluated using a differential flatness parametrization and the differential evolution algorithm with local neighborhood. The Chebyshev and the B-spline approximations are compared. Some preliminary results are reported to compare the performances of the differential evolution with the particle swarm optimization.

Part I

Introductory Elements

1 Optimization and Optimal Control: an Overview

> *Science is organized knowledge.*
>
> **— Immanuel Kant**
> (German philosopher)

Abstract

The outline of the thesis is given, introducing the most important concepts concerning optimization, optimal control and planning of maneuvers for space applications. A brief historical perspective is given so that the reader can understand the context surrounding the objective of this thesis. The reason why an optimal control problem is transformed into a parameter optimization problem is explained and the reasons behind the choice of an heuristic technique are given.

Nomenclature

$\mathbf{X}$	=	Optimization parameters vector	n	=	Optimization parameters number
$\mathbf{S}$	=	Optimization direction	α	=	Optimization step length
$f(\mathbf{X})$	=	NLP cost function	J	=	OCP cost functional
$\bar{f}(\mathbf{X})$	=	Extended NLP cost function	λ	=	Inequality Lagrange multiplier
μ	=	Equality Lagrange multiplier	$g(\mathbf{X})$	=	Inequality constraint
$h(\mathbf{X})$	=	Equality constraint	$\nabla(\cdot)$	=	Gradient operator
L	=	Lagrangian	$\nabla^2(\cdot)$	=	Hessian matrix
G	=	Penalty function	$\boldsymbol{\mathcal{X}}$	=	State admissible set
$\boldsymbol{\mathcal{U}}$	=	Control admissible set	$\boldsymbol{x}(t)$	=	State
$\boldsymbol{u}(t)$	=	Control	E	=	End-point cost functional
F	=	Running cost functional	N_x	=	State dimension
N_u	=	Control dimension	t	=	Time
$\mathbf{b}$	=	Boundary constraint	$\mathbf{p}$	=	Path constraint
$\mathbf{f}$	=	State (differential) equation	$(\cdot)_{0/f}$	=	Initial/final time value
$\mathcal{H}$	=	Hamiltonian	E_a	=	Extended end-point cost functional

1.1 Introduction

Optimality is a fundamental principle that has always led the evolution of nature and human beings, establishing natural lows, ruling biologic behaviors, and con-

ducting social activities . Natural processes that result in the survival or extinction of a species are based on optimization laws, as well as every human being tries to find the optimal solution in decision making problems. In fact, it is a matter of fact that all of us are optimizers as we all make decisions for the purpose of maximizing our quality of life, productivity in time and our welfare, in some way or another. Human history can be analyzed as an ongoing struggle for creating the best possible among many inferior designs. Hence, optimization was, is, and will always be the core requirement of human life, yielding the development of a massive number of innovations, novel techniques, scientific improvements, starting from the early ages of civilization until now (see Ref. [5, 6] for a detailed description of the knowledge fields interested in optimization processes).

Optimization is a recurrent concept in several fields of human knowledge (e.g. economics, mathematics and engineering) and can be described as finding the best solution to a given problem according to an agreed criterion. Such goal of the optimization is usually referred to as cost function, objective function or performance index. The study of optimization problems is also as old as science itself. It is known that the ancient Greek mathematicians solved many optimization problems. For example, around 300 B.C. Euclid proved that a square encloses the greatest area among all possible rectangles with the same total length of four sides [7]. Obviously, the same problem may have different optimal solutions changing the optimality criterion to be satisfied. Planning a maneuver for a body going from a point A to a point B can lead to different thrust profiles if the objective function is the minimum time or the minimum consumption of fuel.

Depending on the object of the optimization process, the denomination of the optimal problem can change. If a finite number of discrete values are searched for, we deal with integer optimization. Continuous optimization, on the contrary, deals with continuous variables. Optimal control is a special class of optimization problems that can be defined within the frame of functional analysis. Consequently, the object of optimal control are functions and the cost function can be referred to as the cost functional as it generally depends on functions.

Two distinct families of approaches can be defined to solve a general optimization problem, i.e. the analytical and the numerical ones. For many practical problems, analytical solutions do not exist and numerical methods must be employed. However, theoretical analysis gives us some necessary conditions that can be used along with numerical methods to solve the optimization problem. Analytical approaches are based on the evaluation of the first and second derivative of the cost function to find the necessary and the sufficient conditions for optimality, respectively. In the field of optimal control, derivatives are substituted by the variations of the cost functional.

The majority of the optimization problems of practical interest are usually constrained such that the solution lies in a set of admissible or feasible solutions. The simple problem of the rectangle with maximum area is constrained to have a fixed perimeter. In this case, the constraint is necessary to obtain a meaningful solution, otherwise the optimal rectangle would have been the one with infinitely-long sides. In these cases, analytical solutions are hard to find but necessary conditions can be obtained. The Karush–Kuhn–Tucker (KKT) conditions apply for continuous optimization problems, whereas the Pontryagin Maximum Principle (PMP) can be employed for optimal control problems (OCPs).

This chapter is intended to explain the relationship between parameter optimization and optimal control. Among the several optimization problems, the OCPs distinguish from the others as they deal with functions. A classical parameter optimization problem (POP), instead, deals with finding a finite set of optimal parameter (usually real or integer numbers). As several OCPs do not have an analytical solution, they are usually transcribed into parameter optimization problems. The way OCPs are transcribed into POPs considerably influences the solutions that can be found, as it will be explained in Chapter 2.

The main result of this thesis is the development of the Inverse-Dynamics Particle Swarm Optimization (IPSO), a numerical optimization technique developed for solving OCPs already employed in Ref. [1, 2, 3, 8, 9]. This name has been chosen in accordance with the following characteristics:

- The differentially flat formulation is employed for the definition of the optimization problem. This method may be also referred to as the *inverse dynamics* approach. The reader can find details about the differentially flat formulation in Chapter 2.

- The Particle Swarm Optimization (PSO) is employed for the numerical solution of the optimization problem. As other heuristic methods, the PSO has specific characteristics which help finding the global minimum when the problem is nonconvex. The reader can find details about the Particle Swarm Optimization in Chapter 3.

The IPSO will be described in details in Chapter 4 and will be applied in Chapters 5, 6 and 8.

The remainder of this chapter is organized as follows. In Sec. 1.3 an outline of the parameter optimization technique is presented, from the deterministic technique to the heuristic and metaheuristic strategies. In Sec. 1.4 the standard problem considered by the optimal control theory is presented, along with the classical necessary conditions for optimality. Conclusions are given in Sec. 1.5.

1.2 Statement of an optimization problem

To simplify the understanding of the differences among the following optimization problems, a common approach for the statement of an optimization problem is introduced. Usually, when a problem is to be solved in order to minimize/maximize a used-defined goal, some features must be identified:

1. The optimization parameters
2. The goal of the optimization
3. The constraints influencing the solution

As a consequence, the general scheme used to state an optimization problem within this thesis is the one reported in Fig. 1.1.

1.3 Optimization: brief outline of the available techniques

To find a numerical solution to OCPs, a numerical optimization technique to search for the optimal set of certain user-defined parameters must be used. In fact, after

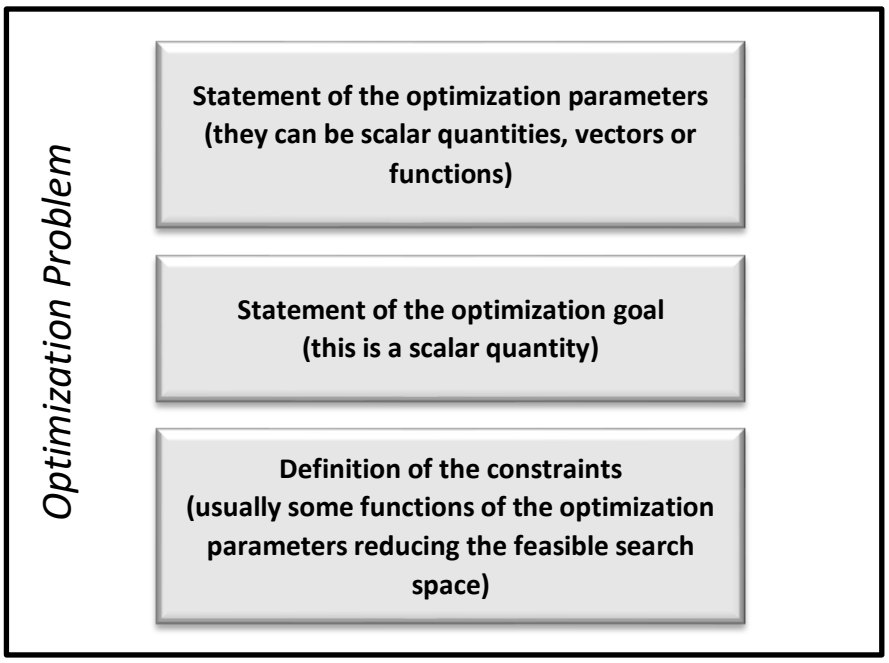


Fig. 1.1: Common steps required to state an optimization problem.

the proper discretization of the problem (which will be properly defined in Chapter 2), OCP is translated into POP [10, 11]. The word "programming" (which has nothing to do with computing programming) is often used to mean planning and/or optimization.

It is far beyond the scope of this work to provide an extensive analysis of the techniques for solving a parameter optimization problem (a detailed description is given in Ref. [12]). OCPs may fall into different parameter optimization problems depending on their characteristics concerning non-linearities issues, constraints, convexities.

The simplest optimization without any constraints is probably the search of the maxima or minima of a function $f(x)$. In 1646 Pierre de Fermat proposed a general approach to compute local maxima/minima points of a differentiable function, that is, setting the derivative of the function to be zero, i.e. $df(x)/dx = 0$. As we know today, this is only a necessary condition for the existence of maxima/minima and the sufficiency condition is related to the second derivative. For one-dimensional problems, many of the classical optimization methods are based on the evaluation of the first derivative and second derivatives, whereas for multi-dimensional problems the gradient and the Hessian matrix are evaluated. Sometimes, this information is obtained numerically after a proper polynomial approximation of the cost function is introduced.

Linear constrained optimization, usually referred to as Linear Programming, is generally solved with the simplex method [13] introduced by G.B. Dantzig in 1947. For this class of problems, usually encountered in the economics field, both the objective function and the constraints are linear. One of the earliest, classic examples using linear programming is described by Dantzig in Ref. [13]. The problem was to find the solution to the special optimal diet problem involving 9 equations and 77 unknowns using hand-operated desk calculators.

Most real-world problems are nonlinear and nonlinear mathematical programming forms an important part of mathematical optimization methods. If the expressions for the objective function and the constraints are fairly simple in terms of the design variables, classical analytical methods of optimization can be used to solve the problem. On the other hand, if the optimization problem involves the objective function and/or constraints that are not stated as explicit functions of the design variables or which are too complicated to manipulate, we cannot solve it by using the classical analytical methods. One-dimensional numerical minimization methods are divided into two big families, the Elimination methods (e.g. Unrestricted search, Exhaustive search, Dichotomous search, Fibonacci method, Golden section method) and the Interpolation methods (quadratic/cubic interpolation, Newton and quasi-

Newton methods). The reader can refer to Chapter 5 of Ref. [12] for further details.

Given the complexity of the problems that will be treated in the chapters concerning the numerical applications (see Part III and Part IV), only numerical techniques for solving multi-dimensional nonlinear problems will be briefly described.

1.3.1 Nonlinear programming for unconstrained problems

Nonlinear programming for unconstrained problems deals with the optimization problem given as

$$\begin{aligned} &\text{Find } \mathbf{X} \in \mathbb{R}^n \\ &\text{minimizing} \quad f(\mathbf{X}) \end{aligned} \tag{1.1}$$

where $f(\mathbf{X}) : \mathbb{R}^n \mapsto \mathbb{R}$. According to Ref. [12], the basic philosophy of most of the numerical methods of nonlinear optimization is to produce a sequence of improved approximations to the optimum according to the following scheme:

1. Start with an initial trial point $\mathbf{X}_1$.

2. Find a suitable direction $\mathbf{S}_i$ (i = 1 to start with) that points in the general direction of the optimum.

3. Find an appropriate step length α_i for movement along the direction $\mathbf{S}_i$.

4. Obtain the new approximation $\mathbf{X}_{i+1}$ as

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \alpha_i \mathbf{S}_i \tag{1.2}$$

5. Test whether $\mathbf{X}_{i+1}$ is optimum. If $\mathbf{X}_{i+1}$ is optimum, stop the procedure. Otherwise, set a new $i = i + 1$ and repeat step (2) onward.

The iterative procedure indicated by Eq. (1.2) is valid for unconstrained as well as constrained optimization problems. Eq. (1.2) indicates that the efficiency of an optimization method depends on the efficiency with which the quantities α_i and $\mathbf{S}_i$ are determined. The methods of finding the optimal step length α_i^* are within the family of one-dimensional numerical minimization methods. In fact, if $f(\mathbf{X})$ is the objective function to be minimized, the problem of determining α_i^* reduces to

finding the value $\alpha_i = \alpha_i^*$ that minimizes $f(\mathbf{X}_{i+1}) = f(\mathbf{X}_i + \alpha_i \mathbf{S}_i) = f(\alpha_i)$ for fixed values of $\mathbf{X}_i$ and $\mathbf{S}_i$.

It is important to note that all the unconstrained minimization methods (1) require an initial point $\mathbf{X}_1$ to start the iterative procedure, and (2) differ from one another only in the method of generating the new point $\mathbf{X}_{i+1}$ (from $\mathbf{X}_i$) and in testing the point $\mathbf{X}_{i+1}$ for optimality.

The methods of finding $\mathbf{S}_i$ may be divided into Direct search methods, which do not require the evaluation of the function derivatives, and Descent methods, which require the derivatives of the function. Among the several methods that falls inside these categories, which are exhaustively described in Chapter 6 of Ref. [12], there are some stochastic technique that can find the global optimum when several local minima exist. For instance, the Random search methods work even if the objective function is discontinuous and nondifferentiable at some of the points. It is noteworthy that this method is heuristic in nature but does not have any well-defined rule guiding the heuristic search. Another direct method is the simplex method (which should not be confused with the simplex method of linear programming) based on the reflection, contraction and expansion operation applied to a simplex, a geometric figure formed by a set of $n + 1$ points in an n-dimensional space. Finally, descent or gradient-based methods (e.g., Newton's Method, Steepest Descent Method, Line Search, Conjugate Gradient Method) are usually characterized by a local convergence and are strongly affected by the initial trial point.

1.3.2 Nonlinear programming for constrained problems

For nonlinear constrained optimization, classical methods provide necessary condition which can be used to find the solution for continuous and differentiable functions. These methods are analytical and make use of the techniques of differential calculus in locating the optimum points. Since some of the practical problems involve objective functions that are not continuous and/or differentiable, the classical optimization techniques have limited scope in practical applications.

For a nonlinear constrained optimization problem (with m inequality constraints and p equality constraints) stated as

$$
\begin{aligned}
&\text{Find } \mathbf{X} \in \mathbb{R}^n \\
&\text{minimizing} \quad f(\mathbf{X}) \\
&\text{subject to} \\
&g_j(\mathbf{X}) \leq 0, \qquad j = 1, ..., m \\
&h_k(\mathbf{X}) = 0, \qquad k = 1, ..., p
\end{aligned}
\tag{1.3}
$$

the necessary KKT conditions are (see Chapter 2 of Ref. [12]):

$$
\nabla f + \sum_{j=1}^{m} \lambda_j \nabla g_j - \sum_{k=1}^{p} \mu_k \nabla h_k = \mathbf{0}, \tag{1.4a}
$$

$$
\lambda_j g_j = 0, \qquad j = 1, ..., m \tag{1.4b}
$$

$$
g_j \leq 0, \qquad j = 1, ..., m \tag{1.4c}
$$

$$
\lambda_j \geq = 0, \qquad j = 1, ..., m \tag{1.4d}
$$

$$
h_k = 0, \qquad k = 1, ..., p. \tag{1.4e}
$$

The parameters λ_j and β_k are called Lagrange multipliers. When $g_j = 0$, the constraint is called active while when $g_j < 0$, the constraint is called nonactive. The optimization problem stated in Eq. (1.7) is called a convex programming problem if the objective function $f(\mathbf{X})$ and the constraint functions are convex. In this case, there will be no relative minima or saddle points and the KKT conditions are both necessary and sufficient for an absolute minimum of at $\mathbf{X}^*$. However, it is often very difficult to ascertain whether the objective and constraint functions involved in a practical engineering problem are convex.

For many engineering applications, the necessary KKT optimality conditions cannot be directly used to find the solution $\mathbf{X}^*$. There are many numerical techniques available for the solution of a constrained nonlinear programming problem. All the methods can be classified into two broad categories: direct methods and indirect methods. In the direct methods, the constraints are handled in an explicit manner, whereas in most of the indirect methods the constrained problem is solved as a sequence of unconstrained minimization problems.

Direct methods

Within the family of direct methods, there are several approaches. Only the Sequential Linear Programming (SLP) and the Sequential Quadratic Programming (SQP)

are therein described. An exhaustive description of the most used approaches can be found in Chapter 7 of Ref. [12].

In the SLP method, the solution of the original nonlinear programming problem is found by solving a series of linear programming problems. Each LP problem is generated by approximating the nonlinear objective and constraint functions using first-order Taylor series expansions about the current design vector, $\mathbf{X}_i$. The resulting LP problem is solved using the simplex method to find the new design vector $\mathbf{X}_{i+1}$. If $\mathbf{X}_{i+1}$ does not satisfy the stated convergence criteria, the problem is re-linearized about the point $\mathbf{X}_{i+1}$ and the procedure is continued until the optimum solution $\mathbf{X}^*$ is found. If the problem is a convex programming problem, the linearized constraints always lie entirely outside the feasible region. Hence the optimum solution of the approximating LP problem, which lies at a vertex of the new feasible region, will lie outside the original feasible region. However, by re-linearizing the problem about the new point and repeating the process, we can achieve convergence to the solution of the original problem in few iterations. When equality and inequality constraints are taken into account, the SLP problem is defined as

$$\begin{aligned} &\text{Find } \mathbf{X} \in \mathbb{R}^n \\ &\text{minimizing } f(\mathbf{X}_i) + \nabla f^T(\mathbf{X} - \mathbf{X}_i) \\ &\text{subject to} \\ &g_j + \nabla g_j^T(\mathbf{X} - \mathbf{X}_i) \leq 0, \qquad j = 1, ..., m \\ &h_k + \nabla h_k^T(\mathbf{X} - \mathbf{X}_i) = 0, \qquad k = 1, ..., p. \end{aligned} \tag{1.5}$$

The SQP is one of the most recently developed and perhaps one of the best methods of optimization. The method has a theoretical basis that is related to 1) the solution of a set of nonlinear equations using Newton's method, and 2) the derivation of simultaneous nonlinear equations applying KKT conditions to the Lagrangian of the constrained optimization problem. When equality and inequality constraints are taken into account, the SQP problem is defined as

$$\begin{aligned} &\text{Find } \mathbf{X} \in \mathbb{R}^n \\ &\text{minimizing} \quad \nabla f^T \Delta\mathbf{X} + \frac{1}{2}\Delta\mathbf{X}^T[\nabla^2 L]\Delta\mathbf{X} \\ &\text{subject to} \\ &g_j + \nabla g_j^T \Delta\mathbf{X} \leq 0, \qquad j = 1, ..., m \\ &h_k + \nabla h_k^T \Delta\mathbf{X} = 0, \qquad k = 1, ..., p \end{aligned} \tag{1.6}$$

where $\Delta\mathbf{X} = (\mathbf{X} - \mathbf{X}_i)$, $[\nabla^2 L]$ is the Hessian matrix of the Lagrangian (also known as Lagrange function) which is given by

$$L = f(\mathbf{X}) + \sum_{j=1}^{m} \lambda_j g_j(\mathbf{X}) + \sum_{k=1}^{p} \mu_k h_k(\mathbf{X}). \tag{1.7}$$

SQP is a globally convergent algorithm, i.e., it will converge to some local solution from any remote starting point (under suitable conditions). Nonlinear optimization problems can have multiple local solutions; the global solution is that local solution corresponding to the least value of $f(\mathbf{X})$. SQP methods, like Newton's method and steepest descent, are only guaranteed to find a local solution of NLP; they should not be confused with algorithms for finding the global solution, which are of an entirely different flavor.

Several numerical techniques have been developed based on SQP implementation. For instance, the software SNOPT (Sparse Nonlinear OPTimizer) is designed for large-scale nonlinear constrained optimization, see Ref. [14]. Note that the solution obtained with SNOPT is generally a local optimum (which may or may not be a global optimum).

Indirect methods and penalty functions

Indirect methods transform the original problem into a sequence of unconstrained minimization problems. There are two main families of indirect methods, the first based on transformation techniques which automatically include the constraints into the problem, the second based on the definition of penalty functions.

Transformation techniques Making a change of variables, a constrained optimization problem may be converted into an unconstrained one (such transformations are not always allowed). For example, consider the constraint represented by $\|\mathbf{X}\|_\infty \leq 1$, where $\|\mathbf{X}\|_\infty$ is the L_∞-norm defined as $\|\mathbf{X}\|_\infty = \max\{|x_i(t)| : i = 1, ..., n\}$ and x_i is the ith component of the design vector $\mathbf{X}$. In this case, defined the new design variable $y_i \in \mathbb{R}$, one of the following transformation can be used:

$$x_i = \sin^2 y_i, \tag{1.8a}$$

$$x_i = \cos^2 y_i, \tag{1.8b}$$

$$x_i = \frac{e^{y_i}}{e^{y_i} + e^{-y_i}}, \tag{1.8c}$$

$$x_i = \frac{y_i^2}{1 + y_i^2}. \tag{1.8d}$$

Note the following aspects of transformation techniques:

1. The constraints $g_j(\mathbf{X})$ have to be very simple functions of x_i.

2. For certain constraints it may not be possible to find the necessary transformation.

3. If it is not possible to eliminate all the constraints by making a change of variables, it may be better not to use the transformation at all. The partial transformation may sometimes produce a distorted objective function which might be more difficult to minimize than the original function.

Penalty functions The performance index can be properly modified to take into account the alteration of the search space induced by the constraints. All the constraints are treated as inequalities. Let us suppose to have p equality constraints and m inequality constraints. Accordingly, the generic equality constraint $h_k(\mathbf{X}) = 0$, $k = 1, ..., p$, is *relaxed* and treated as $|h_k(\mathbf{X})| - \Delta_k \leq 0$, defining the new inequality constraint $g_{k+m}(\mathbf{X}) \leq 0$. Note that g_{k+m} converge to h_k as $\Delta_k \to 0$.

Penalty function methods transform the basic optimization problem into alternative formulations such that numerical solutions are sought by solving a sequence of unconstrained minimization problems. Let the basic optimization problem, with inequality constraints, be of the form:

$$\begin{gathered} \text{Find } \mathbf{X} \in \mathbb{R}^n \\ \text{minimizing } f(\mathbf{X}) \\ \text{subject to} \\ g_j(\mathbf{X}) \leq 0, \quad j = 1, ..., m + p \end{gathered} \tag{1.9}$$

This problem is converted into an unconstrained minimization problem by defining the problem an extended cost function $\bar{f}$ (or performance index) as

$$\begin{gathered} \text{Find } \mathbf{X} \in \mathbb{R}^n \\ \text{minimizing } \bar{f}_k = \bar{f}(\mathbf{X}, r_k) = f(\mathbf{X}) + r_k \sum_{j=1}^{m} G_j[g_j(\mathbf{X})] \end{gathered} \tag{1.10}$$

where G_j is some function of the constraint g_j, and r_k is a positive constant known as the penalty parameter. The second term on the right side of Eq. (1.10) is called the penalty term and the generic term G_j is the penalty function associated to the

constraint $g_j(\mathbf{X})$. If the unconstrained minimization of the $\bar{f}$ function is repeated for a sequence of values of the penalty parameter $r_k,\ (k = 1, 2, ...)$, the solution may be brought to converge to that of the original problem stated in Eq. (1.9). This is the reason why the penalty function methods are also known as sequential unconstrained minimization techniques (SUMTs). The penalty function formulations for inequality constrained problems can be divided into two categories: interior and exterior methods. In the interior formulations, some popularly used forms of G_j are given by

$$G_j = -\frac{1}{g_j(\mathbf{X})}, \tag{1.11}$$

$$G_j = \log\,(-g_j(\mathbf{X})). \tag{1.12}$$

Some commonly used forms of the function G_j in the case of exterior penalty function formulations are

$$G_j = \max[0, g_j(\mathbf{X})], \tag{1.13}$$

$$G_j = (\max[0, g_j(\mathbf{X})])^2. \tag{1.14}$$

In the interior methods, the unconstrained minima of $\bar{f}_k$ all lie in the feasible region and converge to the solution of Eq. (1.9) as r_k is varied in a particular manner. In the exterior methods, the unconstrained minima of $\bar{f}_k$ all lie in the infeasible region and converge to the desired solution from the outside as r_k is changed in a specified manner. The convergence of the unconstrained minima of $\bar{f}_k$ is illustrated in Fig. 1.2 for the simple problem

$$\begin{gathered} \text{Find } X = \{x_1\} \text{ which minimizes } f(X) = \alpha x_1 \\ \text{subject to} \\ g_1(X) = \beta - x_1 \leq 0 \end{gathered} \tag{1.15}$$

It can be seen from Fig. 1.2a that the unconstrained minima of $\bar{f}(X, r_k)$ converge to the optimum point X^* as the parameter r_k is increased sequentially. On the other hand, the interior method shown in Fig. 1.2b gives convergence as the parameter r_k is decreased sequentially.

There are several reasons for the appeal of the penalty function formulations. One main reason, which can be observed from Fig. 1.2, is that the sequential nature of the method allows a gradual or sequential approach to criticality of the constraints. In addition, the sequential process permits a graded approximation to be used in analysis of the system. This means that if the evaluation of f and g_j (and hence $\bar{f}(X, r_k)$) for any specified design vector X is computationally very difficult, we can use coarse approximations during the early stages of optimization (when the unconstrained minima of $\bar{f}_k$ are far away from the optimum) and finer or more

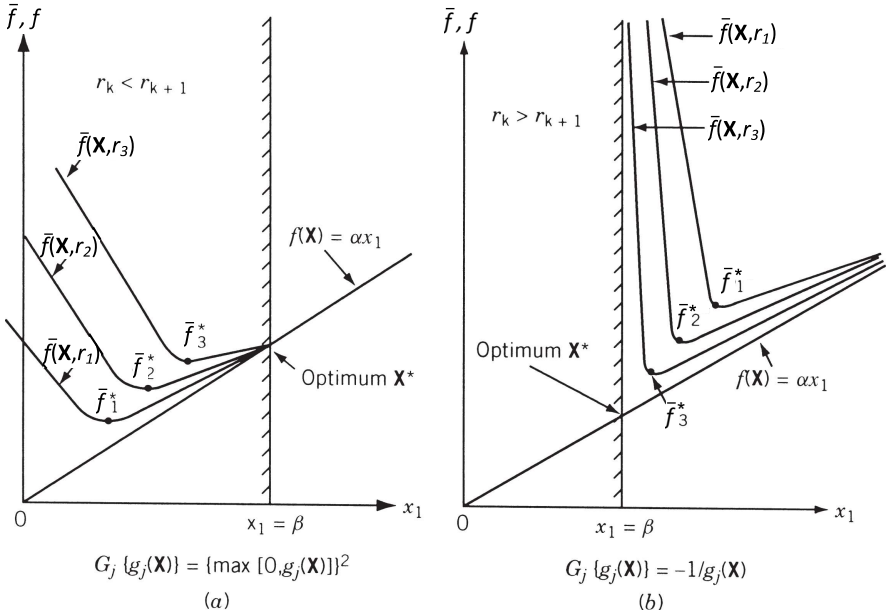


Fig. 1.2: Penalty function methods: (a) exterior method; (b) interior method (adapted from Ref. [12]).

detailed analysis approximation during the final stages of optimization. Another reason is that the algorithms for the unconstrained minimization of rather arbitrary functions are well studied and generally are quite reliable.

Among the penalty function methods, the class of by the interior-point methods has received particular attention [15]. Commercial softwares have been developed basing on interior-point implementations, such as IPOPT (Interior Point OPTimizer, pronounced eye-pea-Opt) which is described in Ref. [16]. Note that these software are generally designed to find local solutions of nonlinear constrained optimization problems.

1.3.3 Heuristic and metaheuristic programming

Most traditional optimization paradigms move from one point in the decision hyperspace to another using some deterministic rule. One of the drawbacks of this approach is the likelihood of getting stuck at a local optimum.

Stochastic algorithms generally overcome this issue, as they are usually intended for searching the global optimal solution for problem with multiple local minima (usually nonconvex problems). For stochastic algorithms, we have in general two types: heuristic and metaheuristic, though their difference is small.

A stochastic algorithm is often the second-best way to solve a problem. Classical methods such as linear programming should often be tried first, as should customized approaches that take full advantage of knowledge about the problem. However, classical and customized approaches are often not feasible, while heuristic and metaheuristic paradigms are usable in a vast number of situations. In fact, the real strength of heuristic and metaheuristic paradigms is that they are generally quite robust. In this field, robustness means that an algorithm can be used to solve many problems, and even many kinds of problems, with a minimum amount of special adjustments to account for special qualities of a particular problem. Typically a stochastic algorithm requires specification of the length of the problem solution vectors, some details of their encoding, and an evaluation function—the rest of the program does not need to be changed.

Heuristics

A heuristic algorithm is defined in Ref. [17] as a technique consisting of a rule (or a set of rules) which seeks (and hopefully finds) good solutions at a reasonable computational cost. A heuristic is approximate in the sense that it provides (hopefully) a good solution for relatively little effort, but it does not guarantee optimality. Heuristics provide simple means of indicating which among several alternatives seems to be best. That is, heuristics are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises between two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices. A heuristic may be a rule of thumb that is used to guide one's action.

According to Ref. [7], heuristic means "to find" or "to discover by trial and error". Quality solutions to a tough optimization problem can be found in a reasonable amount of time, but there is no guarantee that optimal solutions are reached. It is expected that these algorithms work most of the time, but not all the time. This is usually good enough when we do not necessarily want the best solutions but rather good solutions which are easily reachable.

Random search methods for unconstrained minimization (described in Sec. 1.3.1) can be used, with minor modifications, to solve a constrained optimization problem. This approach generates a trial design vector using one random number for each design variable. If any constraint is violated (the equality constraints are considered satisfied whenever their magnitudes lie within a specified tolerance), new trial vectors are generated and verified. Once a trial vector satisfies all the constraints, it is kept as the best design vector if it gives a reduced objective function value

compared to the previous best available design vector. Otherwise, new trial design vectors are generated until a specified maximum number of trial design vectors have been tried. Further details are given in Ref. [18].

Metaheuristics

Further development over the heuristic algorithms is the so-called metaheuristic algorithms. According to Ref. [7], *meta* means "beyond" or "higher level", and they generally perform better than simple heuristics. In addition, all metaheuristic algorithms use certain trade-off of randomization and local search. It is worth pointing out that no agreed definitions of heuristics and metaheuristics exist in the literature, i.e. some use "heuristics" and "metaheuristics" interchangeably. However, recent trends tend to name all stochastic algorithms with randomization and local search as metaheuristic. Randomization provides a good way to move away from local search to a search on the global scale. Therefore, almost all metaheuristic algorithms intend to be suitable for global optimization.

In Ref. [17], a metaheuristic algorithm is defined as a top-level strategy that guides an underlying heuristic solving a given problem. Metaheuristics, in their modern forms, are based on a variety of interpretations of what may be called intelligent search, where the term "intelligent search" has been made prominent by Pearl in Ref. [19]. In that sense we may also consider the following definition in Ref. [20]: "A metaheuristic is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for *exploring* and *exploiting* the search spaces using learning strategies to structure information in order to find efficiently near-optimal solutions". To summarize, the following definition seems to be most appropriate: "A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration." (page ix in Ref. [21]).

A slightly different definition of a metaheuristic algorithm is given in Ref. [22], where it is defined as an algorithm designed to solve approximately a wide range of hard optimization problems without having to deeply adapt to each problem. Indeed, the Greek prefix *meta*, present in the name, is used to indicate that these algorithms are higher level heuristics, in contrast with problem-specific heuristics. Metaheuristics are generally applied to problems for which there is no satisfactory problem-specific algorithm to solve them.

As reported in Ref. [22], almost all metaheuristic algorithms share the following characteristics:

- they are nature-inspired, based on some principles from physics, biology or ethology;
- they make use of stochastic components (involving random variables);
- they do not use the gradient or Hessian matrix of the objective function;
- they have several parameters that need to be fitted to the problem at hand.

Exploration and exploitation

As reported in Ref. [22], a metaheuristic approach will be successful on a given optimization problem if it can provide a balance between the *exploration* (or diversification) and the *exploitation* (or intensification). Exploration is usually associated with global search ability, whereas exploitation is related to local search. In further details, Ref. [23] states that "exploration is the process of visiting entirely new regions of a search space, whilst exploitation is the process of visiting those regions of a search space within the neighborhood of previously visited points. In order to be successful, a search algorithm needs to establish a good ratio between exploration and exploitation".

Exploitation is needed to identify parts of the search space with high quality solutions, and it is important to intensify the search in some promising areas of the accumulated search experience. The main differences between the existing metaheuristics concern the particular way in which they try to achieve this balance. Many classification criteria may be used for metaheuristics. This may be illustrated by considering the classification of metaheuristics in terms of their features with respect to different aspects concerning the search path they follow, the use of memory, the kind of neighborhood exploration used or the number of current solutions carried from one iteration to the next.

Classification of metaheuristic algorithms

Following the classification reported in Ref. [22], metaheuristic algorithms differentiates between single-solution based metaheuristics (also known as trajectory-based metaheuristics) and population-based metaheuristics. Roughly speaking, single-solution based metaheuristics (e.g., simulated annealing) are more exploitation oriented whereas population-based metaheuristics (e.g., evolutionary computation algorithms) are more exploration oriented.

Among the single-solution based metaheuristic approaches there are simulated annealing, Tabu search, GRASP method, variable neighborhood search, guided local search, iterated local search (refer to Ref. [22] for details). For example, simulated annealing uses a single agent or solution which moves through the design space or search space in a piecewise style. A better move or solution is always accepted, while a not-so-good move can be accepted with certain probability. The steps or moves trace a trajectory in the search space, with a non-zero probability that this trajectory can reach the global optimum.

Population-based metaheuristic algorithms may be subdivided into:

- Evolutionary Computation, EC (e.g., genetic algorithm, evolution strategy, evolutionary programming, genetic programming, differential evolution). A description of the most important approaches is reported in Ref. [22].

- Swarm Intelligence, SI (e.g., particle swarm optimization, ant colony optimization, bacterial foraging optimization algorithm). The reader can refer to Ref. [24, 22] for a summary of the approaches within this class.

A detailed description of the Particle Swarm Optimization (PSO) will be given in Chapter 3. Moreover, the Differential Evolution (DE) algorithm will be exploited in Chapter 9.

Most metaheuristic algorithms are nature (biology, bio)-inspired as they have been developed based on some abstraction of nature. Nature has evolved over millions of years and has found perfect solutions to almost all the faced problems. We can thus learn the success of problem-solving from nature and develop nature-inspired heuristic algorithms.

Two major components of any metaheuristic algorithms are 1) selection of the best solutions and 2) randomization. The selection of the best ensures that the solutions will converge to the optimality, while the randomization avoids the solutions being trapped at local optima and, at the same, increase the diversity of the solutions. The selection in here intended as recognition of the best solutions. For most EC techniques, selection is related to discarding (from future investigation) non-selected individuals. The good combination of these two components will usually ensure that the global optimality is achievable. Heuristic algorithms can be classified in many ways.

Population-based paradigms are generally characterized by

1. utilizing a population of points (potential solutions) in their search.

2. using direct fitness information instead of function derivatives or other related knowledge.

3. using probabilistic, rather than deterministic, transition rules.

In addition, EC implementations sometimes encode the parameters in binary or other symbols, rather than working with the parameters themselves.

Population-based metaheuristic algorithms start with a population of points (hyperspace vectors). They typically generate a new population with the same number of members each epoch, or generation. Thus, many maxima or minima can be explored simultaneously, lowering the probability of getting stuck. Operators such as crossover and mutation typical of evolutionary computation techniques, or the PSO velocity, effectively enhance this parallel search capability. The main important differences between evolutionary computation and swarm intelligence lye in the way the population of solutions is modified through successive iterations. Selection of individuals for reproduction to constitute a new population (often called a new generation) is usually based upon fitness values. If the goal is the minimization of the fitness, the lower the fitness, the more likely it is that the individual will be selected for the new generation. Some paradigms that are sometimes considered evolutionary, such as particle swarm optimization, can retain all population members from epoch to epoch.

Regardless of the implemented paradigm, population based algorithms often follow a similar procedure:

1. Initialize the population.

2. Calculate the fitness for each individual in the population.

3. Produce a new population basing on some rules that strictly depend on the fitness of each individual.

4. Loop to step 2 until some condition is met.

Initialization is most commonly done by seeding the population with random values. When the parameters are represented by binary strings, this simply means generating random strings of ones and zeros (with a uniform probability for each value) of the fixed length described earlier. It is sometimes feasible to seed the population with promising values, known to be in the hyperspace region relatively close to the optimum. The total number of individuals chosen to make up the population is both problem and paradigm dependent, but is often in the range of a few dozen to a few hundred. The fitness value is often proportional to the output

value of the function being optimized, though it may also be derived from some combination of a number of function outputs.

Termination of the algorithm is usually based either on achieving a population member with some specified fitness or on having run the algorithm for a given number of generations. In many, if not most, cases, a global optimum exists at one point in the decision hyperspace. Frequently there are very good local optima as well. For these and other reasons, the bottom line is that it is often unreasonable to expect any optimization method to find a global optimum (even if it exists) within a finite time. The best that can be hoped for is to find near-optimum solutions and to hope that the time it takes to find them increases less than exponentially with the number of variables.

1.4 Optimal control problems

In this section a general continuous-time OCP is defined and the first-order necessary optimality conditions for that problem are derived using the calculus of variations. Pontryagin's principle, which is used to solve for the optimal control in some special cases, is also discussed.

1.4.1 Continuous-time optimal control problem

If an optimization problem involves the minimization (or maximization) of a functional subject to the constraints of the same type, the decision variable will not be a number, but it will be a function. The calculus of variations can be used to solve this type of optimization problems. The main aim of the calculus of variations is to find a function that makes the integral stationary, making the value of the integral a local maximum or minimum. For example, in mechanics we may want to find the shape $y(x)$ of a rope or chain when suspended under its own weight from two fixed points. In this case, the calculus of variations provides a method for finding the function $y(x)$ so that the curve $y(x)$ minimizes the gravitational potential energy of the hanging rope system.

When the objective is an integral and the constraints are differential equation, the optimization problem becomes an OCP. Optimal control is an important branch of optimization and control research, especially in engineering design and economics. For example, in order to reach from point A to B on a road $u(t)$, we can vary the speed $a(t)$ of a car so as to minimize the fuel consumption is an OCP. Similarly, to design a railway path on a hilly landscape with the constraint of slope or gradient so as to minimize the distance between any two stations also requires optimal control. OCPs involve two types of variables: the control and state variables, which are

related to each other by a set of differential equations. Optimal control theory can be used for solving such problems.

Here and in the following chapters we will only deal with free final time problems, whereas the initial time will be considered as a fixed value (usually equal to zero). Given $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^{N_x}$ and $\boldsymbol{u} \in \mathbb{U} \subset \mathbb{R}^{N_u}$ (N_x and N_u are scalar problem-dependent parameters), the standard OCP for a dynamical system subject to the dynamics

$$\dot{\boldsymbol{x}}(t) = \mathbf{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \tag{1.16}$$

with boundary constraints (defined as equality constraints)

$$\mathbf{b}(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f) = \mathbf{0} \tag{1.17}$$

and state-control path constraints given by the inequality constraint

$$\mathbf{p}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq \mathbf{0}, \tag{1.18}$$

may be stated as:

$$\min J[\boldsymbol{x}(t), \boldsymbol{u}(t), t_f] = E(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} F(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, dt \tag{1.19}$$

subject to Eq. (1.16), (1.17) and (1.18). It is assumed that $\mathbf{f} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R} \to \mathbb{R}^{N_x}$, $\mathbf{b} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_x} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{N_e}$ and $\mathbf{p} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \to \mathbb{R}^{N_p}$. As well as N_x and N_u, also N_e and N_p are problem dependent. All these functions are assumed to be continuously differentiable with respect to their arguments. In Eq. (1.19), $E : \mathbb{R}^{N_x} \times \mathbb{R}^{N_x} \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ is the terminal cost function, $F : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R} \mapsto \mathbb{R}$ is the running cost function.

Following the general rules of Sec. 1.2 for the statement of an optimization problem, an optimal control problem is written as

$$\begin{gathered}
\text{Find } \boldsymbol{x}(t) : t \to \mathbb{X} \subset \mathbb{R}^{N_x}, \boldsymbol{u}(t) : t \to \mathbb{U} \in \mathbb{R}^{N_u}, t_f \in \mathbb{R} \\
\text{minimizing} \\
J[\boldsymbol{x}(t), \boldsymbol{u}(t), t_0, t_f] = E(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f) + \int_{t_0}^{t_f} F(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, dt \\
\text{subject to, } \forall t \in [t_0, t_f] \\
\text{Dynamics constraints:} \quad \dot{\boldsymbol{x}}(t) = \mathbf{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \\
\text{Boundary constraints:} \quad \mathbf{b}(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f) = \mathbf{0}, \\
\text{Path Constraints:} \quad \mathbf{p}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq \mathbf{0}.
\end{gathered} \tag{1.20}$$

When stating an optimal control problem, state and control should be defined as *functions*. Usually, the state is a continuous and piecewise differentiable function whereas the control is piecewise continuous function. However, since this thesis mainly deal with numerical solutions to optimal control problems, we are not interested in the mathematical properties of the state and control functions. Instead, the dimension of state and control are important from the numerical point of view. Accordingly, state and control functions are defined by means of their vectorial values corresponding to the generic time instant t. As reported in Eq. (1.20), such vectors live in the state space $\mathbb{X}$ and in the control space $\mathbb{U}$, respectively.

The solution of the minimization problem in Eq. (1.19) is given in several standard textbooks (see Ref. [25, 26]). The solution reported below is taken from Ref. [27]. As usual, let us define the Hamiltonian as

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) = \boldsymbol{\lambda}^T(t)\mathbf{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + F(\boldsymbol{x}(t), \boldsymbol{u}(t), t). \tag{1.21}$$

The optimality necessary condition are stated by the Pontryagin's Maximum Principle which is described in Sec. 1.4.2. In presence of path constraint, the Lagrangian of the Hamiltonian may be introduced,

$$L(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) + \boldsymbol{\mu}^T\mathbf{p}(t)(\boldsymbol{x}(t), \boldsymbol{u}(t)). \tag{1.22}$$

Note that Eq. (1.22) is the functional counterpart of Eq. (1.7). Finally, it can be shown that the necessary conditions are stated as

$$\dot{\boldsymbol{x}}(t) = \frac{\partial L(t)}{\partial \boldsymbol{\lambda}} \tag{1.23}$$

$$\dot{\boldsymbol{\lambda}}(t) = -\frac{\partial L(t)}{\partial \boldsymbol{x}} \tag{1.24}$$

$$\frac{\partial L}{\partial \boldsymbol{u}} = \mathbf{0} \tag{1.25}$$

$$\{\boldsymbol{\lambda}(t_0), \boldsymbol{\lambda}(t_f)\} = \left\{ -\frac{\partial E_a}{\partial \boldsymbol{x}(t_0)}, \frac{\partial E_a}{\partial \boldsymbol{x}(t_f)} \right\} \tag{1.26}$$

$$\{\mathcal{H}(t_0), \mathcal{H}(t_f)\} = \left\{ -\frac{\partial E_a}{\partial t_0}, \frac{\partial E_a}{\partial t_f} \right\} \tag{1.27}$$

$$\boldsymbol{\mu}^T(t)\mathbf{p}(t) = \mathbf{0} \ , \ \mathbf{p} \leq \mathbf{0} \ , \ \boldsymbol{\mu} \geq \mathbf{0} \tag{1.28}$$

where the extended end-point cost functional E_a is defined as

$$E_a(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f, \boldsymbol{\nu}) = E(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f) + \boldsymbol{\nu}^T\mathbf{b}(E_a(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f). \tag{1.29}$$

If the path constraint in Eq. (1.18) is independent of the control (i.e. a pure state constraint) then the costate $\boldsymbol{\lambda}(t)$ must satisfy the jump condition reported in [28].

1.4.2 Pontryagin's principle

The Pontryagin's Minimum Principle (PMP) (originally introduced as Pontryagin's Maximum Principle) is given as

$$\mathcal{H}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t), t) \leq \mathcal{H}(\boldsymbol{x}^*(t), \boldsymbol{u}(t), \boldsymbol{\lambda}^*(t), t). \tag{1.30}$$

Note that, in Eq. (1.30), $(\ )^*$ stands for a function which is considered optimal with regard to the imposed performance measure. Eq. (1.30) has been firstly introduced in Ref. [29] and states that the optimal control $\boldsymbol{u}^*$ is such to minimize the Hamiltonian. In the case the control region is unbounded, the necessary condition in Eq. (1.30) can be expresed as

$$\frac{\partial \mathcal{H}}{\partial \boldsymbol{u}}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t), t) = \boldsymbol{0}. \tag{1.31}$$

If Eq. (1.31) is satisfied, and the matrix

$$\frac{\partial^2 \mathcal{H}}{\partial \boldsymbol{u}^2}(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t), \boldsymbol{\lambda}^*(t), t) \tag{1.32}$$

is positive definite, this is sufficient to guarantee that $\boldsymbol{u}^*(t)$ causes $\mathcal{H}$ to be a *local* minimum. However, if $\mathcal{H}$ is a quadratic form in $\boldsymbol{u}(t)$, then Eqs. (1.31)-(1.32) guarantee that $\boldsymbol{u}^*(t)$ is a *global* minimum.

During the 1960s, the Maximum Principle came to be the primary tool for solving OCPs. Flight trajectory optimization continued to be the main application and the driving force in the field. PMP transforms the OCP into a two-point boundary-value problem (TPBVP, see Ref. [30]). For most cases, other than simple textbook problems, the solution procedure poses a serious obstacle for implementation.

Let us solve a minimum-time problem for a control-affine dynamical system employing PMP. The dynamical system is described by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}\left(t\right)\right) + \boldsymbol{g}\left(\boldsymbol{x}\left(t\right)\right)\boldsymbol{u}\left(t\right) , \tag{1.33}$$

where $\boldsymbol{X}(t) : t \to \mathbb{R}^{N_X}$ is the state function (which has to satisfy some continuity properties up to the first time derivative) and $\boldsymbol{U}(t) : t \to \mathbb{R}^{N_U}$ is the external control function (which does not necessarily have to be continuous). N_X and N_U are the state and the control dimensions, respectively. The control $\boldsymbol{U}(t)$ is supposed to lye in an admissible region $[\boldsymbol{u}_{min}, \boldsymbol{u}_{max}]$. Note that Eq. (7.1) may be non-linear in the state (through the term $\boldsymbol{f}(\boldsymbol{x}(t))$) but is affine in the control. The operator $\boldsymbol{g}(\boldsymbol{x}(t))$

may be non-linearly dependent on the state and it is expressed by a $N_x \times N_u$ matrix. The dynamical system may also be affected by equality boundary conditions

$$\mathbf{b}(\boldsymbol{x}_0, \dot{\boldsymbol{x}}_0, \boldsymbol{x}_f, \dot{\boldsymbol{x}}_f, t_0, t_f) = \mathbf{0} \tag{1.34}$$

and, $\forall t \in [t_0, t_f]$, by inequality path constraints

$$\mathbf{p}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t) \leq \mathbf{0}\,. \tag{1.35}$$

Note that both Eq. (7.2) and (7.3) do not depend on $\boldsymbol{u}(t)$.

The performance measure to be minimized is

$$J = t_f\,. \tag{1.36}$$

The theory for this class of problems is known and may be found in Ref. [25, 26]. Here, we are only interested in giving the justification and the definition of the typical bang-bang structure of the external control. Hence, let us define the Hamiltonian $\mathcal{H}$ as

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = \boldsymbol{\lambda}^T(t)(\boldsymbol{f}\left(\boldsymbol{x}\left(t\right)\right) + \boldsymbol{g}\left(\boldsymbol{x}\left(t\right)\right)\boldsymbol{u}\left(t\right))\,, \tag{1.37}$$

where $\boldsymbol{\lambda}(t)$ is the costate function. When the path constraint $\boldsymbol{p}$ does not depend on $\boldsymbol{u}(t)$, the Pontryagin Maximum Principle [29] holds and Eq. (1.30) must be true $\forall t \in [t_0, t_f]$. It follows that, if $\boldsymbol{g}_i(\boldsymbol{x})$ is the i^{th} column of $\boldsymbol{g}(\boldsymbol{x})$, than the i^{th} component $u_i(t)$ of $\boldsymbol{u}(t)$ is given by

$$u_i(t) = \begin{cases} u_{max} & \text{if } \boldsymbol{\lambda}^T(t)\boldsymbol{g}_i(\boldsymbol{x}(t)) < 0, \\ u_{min} & \text{if } \boldsymbol{\lambda}^T(t)\boldsymbol{g}_i(\boldsymbol{x}(t)) > 0, \\ \text{Undetermined} & \text{if } \boldsymbol{\lambda}^T(t)\boldsymbol{g}_i(\boldsymbol{x}(t)) = 0\,. \end{cases} \tag{1.38}$$

In the following, we make the assumption that the optimal solution is not affected by singular arcs with undetermined control. As a consequence, the optimal control structure is bang-bang, i.e. $\boldsymbol{u}(t)$ does only take extremal values, u_{min} or u_{max}, $\forall t \in [t_0, t_f]$. Moreover, after the appropriate normalization, we can state that $u_{min} = -1$ and $u_{max} = +1$.

1.5 Endnotes

In this introductory chapter, the most important features of an optimal control problems have been outlined. The relationship between optimization and optimal control problems has been described, stating that the latter is a sub-family of the general optimization problems as it deals with functions. However, numerical

solutions transcribe the initial optimal control problem into a parameter optimization problem, so that a numerical technique for nonlinear programming is required. In this thesis, the swarm intelligence will be employed, obtained solutions for complex problems by means of the Particle Swarm Optimization. Accordingly, after a brief explanation of classical deterministic techniques, the general features of the metaheuristic approaches have been described, underlining the ability to search for the global optimal solution.

2 Numerical Transcription of an Optimal Control Problem

> *The formulation of the problem is often more essential than its solution, which may be merely a matter of mathematical or experimental skill.*
>
> — **Albert Einstein**
> (German physicist)

Abstract

The numerical transcription of an optimal control problem is a necessary step when an approximated solution is searched for. An essential role is played on the way the optimal control problem is transcribed. Different definitions of the independent variables, different number of optimization variables and different numerical optimization techniques can make an optimization problem attain several solutions. Also, the same solution can be obtained with different computational efforts depending on the numerical transcription. This chapter introduces the reader to the differential flatness formulation of an optimal control problem, reporting the general advantages and disadvantages of this approach.

Nomenclature

$\boldsymbol{x}(t)$	=	State	$\boldsymbol{u}(t)$	=	Control
N_x	=	State dimension	N_u	=	Control dimension
E	=	End-point cost functional	F	=	Running cost functional
J	=	Performance index	$\mathbf{b}$	=	BC function
$\mathbf{p}$	=	State-control PC function	$(\cdot)^N$	=	Numerical approximation
ϕ	=	Polynomial basis	$\boldsymbol{\mathcal{F}}$	=	Differential inclusion map
$N_{\mathcal{P}}$	=	Number of polynomial coefficients	$\boldsymbol{y}$	=	Flat output
$(\cdot)_{0/f}$	=	Initial/final time value	$\boldsymbol{\mathcal{X}}$	=	State admissible set
$\boldsymbol{\mathcal{U}}$	=	Control admissible set	q_1, q_2	=	Numerical example state
m, g	=	Numerical example parameters	I_1, I_2	=	Numerical example parameters
$\tilde{(\cdot)}$	=	Approximation coefficient	N_T	=	Discretization points
k, l	=	Numerical example parameters	t	=	Time
$\mathbb{X}$	=	State space	$\mathbb{U}$	=	Control space
$\mathbb{Y}$	=	Flat output space	$\mathbf{f}$	=	State dynamics function

2.1 Introduction

Every time an optimal control problem (OCP) is to be solved, most of the time to get the solution must be spent in understanding the problem and finding the proper transcription. The way the problem is formulated strongly affect the results that can be obtained or the computational effort required for the evaluation of the optimal control policy.

The optimal control theory typically employed in engineering fields is based upon two important quantities, the state and the control [25]. The former describes the behavior of the dynamical system whereas the latter represents the input that dictates the evolution of the system. This distinction is employed in classical studies concerning optimal control. For instance, PMP provides the necessary conditions an optimal control policy should have in order to minimize/maximize the cost function, as described in Sec. 1.4.2. The limits of this theory lie in the fact that one can state the necessary optimality conditions for some dynamical systems, but the control policy may be found only in very special and simplified cases. In Sec. 1.4.2 an example of application has been reported. However, as one can see, we only have discovered the extremal nature of the optimal control policy. We can infer the number of switches and the values of the control only analyzing the problem, and it is possible only for simple, academic cases.

For a great number of dynamical systems, however, different formulations may be used which involve fewer independent quantities to describe the problem. The idea of the inverse dynamics approach is very common in mathematics where it is often referred to as differential inclusion (DI). Many works related to this topic have been published (e.g., see Ref. [31]). Here, the classical PMP has been revised and extended to more complicated dynamical systems.

Many of the applications reported in this thesis employ the differential flatness (DF) formulation [32] which is based on the identification of a minimum number of independent flat outputs that can completely describe and solve an OCP. The advantage of reducing the number of unknowns, however, is usually coupled to other undesirable numerical properties. Using collocation-based methods, for instance, the differential inclusion formulation can worsen the tractability of the state equations [33]. In the same way, for generic Bolza problems with running cost and terminal cost, the differential flatness formulation may transform an initial convex cost functional into a nonconvex one [34].

The convexity problem plays a crucial role both for the mathematical treatment of OCPs and for numerical applications [35]. Usually, for nonconvex problems, some related relaxed OCPs are introduced to simplify the study. The *convexification* tech-

nique is quite usual both for mathematical[36, 37] and engineering application[38, 39] and allows one to approximate the original nonconvex problem with another, similar one recovering the convexity property. In fact, several numerical techniques can easily get stuck over local minima introduced by nonconvex problems. For instance, applying a pseudospectral approach [40] based on a sequential quadratic programming solver[41], it has been shown that a nonconvex cost functional can create severe convergence problems [34].

Nonconvexity issues for OCPs can be due to [42]: 1) nonconvex cost, 2) nonlinear state dynamics, 3) nonconvex state constraints and 4) nonconvex control constraints. Such nonconvexity problems may be either due to the nature of the problem or they may arise depending on the parametrization employed for the formulation of the optimization problem. Due to the implementation of a differential flatness approach, the problem that will be taken into account possesses all the above issues.

In the following sections several abstract, mathematical formulations of the Bolza problem are given in order to understand the differentially flat formulation employed in the IPSO and compare it to other typical formulations. The example taken into account will clarify how the formulation impacts the practical solution. We will focus on autonomous dynamical systems where the time variable does not appear explicitly in the equations. These systems are often encountered in engineering problems and the reported example is taken from this class of problems.

The chapter is organized as follows. In Sec. 2.2 an introduction to the relationships between formulation and transcription of OCPs is given. In Sec. 2.3 an example problem is introduced to make it easier understand the proposed OCP formulations. In Secs. 2.4, 2.5 and 2.6 the state-control, differential inclusion and differential flatness formulations are described, respectively. Finally, final remarks are given in Sec. 2.7.

2.2 Formulation of the optimal control problem

Three different possible ways to set and solve a Bolza Optimal Control Problem (BOCP) are described, where the last one is the DF approach. On the one hand we will give the required mathematical formulation, without entering into the details of the necessary optimality conditions that can be derived. On the other hand, we will describe the necessary transcription required when solving a BOCP. In fact, in the mathematical formulation of the BOCP, the solution lives in some infinite-dimension function space (e.g., the absolutely continuous functions or the piecewise continuous functions). The numerical transcription of the BOCP transforms the infinite-dimension problem into a finite-dimension problem where only a finite

number of parameters must be found. With this regard, we can consider that each unknown function is written as a polynomial completely identified by $N_{\mathcal{P}}$ unknown parameters.

The classification of the OCP transcriptions has been considered in several works. The classical distinction is between *direct methods* and *indirect methods*. In Ref. [43], for example, it is reported that the indirect methods are based on the calculus of variations or the Pontryagin's maximum principle, i.e. the conditions described in Sec. 1.4. To obtain solutions from these necessary conditions we may use 1) methods which are based on the special structure of these necessary conditions, e.g. so-called gradient methods, or 2) multiple shooting methods, which require rather good initial approximations of the optimal trajectory and an a-priori knowledge of the switching structure of the constraints. All in all, the user must have a deep insight into the physical and mathematical nature of the optimization problem. In direct approaches OCP is transformed into a nonlinear programming problem. For example, this can be done with a so-called direct shooting method through a parameterization of the controls. For this we choose $\boldsymbol{u}(t)$ from a finite dimensional space of control functions and use explicit numerical integration to satisfy the differential equations for the state dynamics. Inside this class, one of the most used approach is the collocation method. This method was firstly introduced by Hargraves and Paris in Ref. [44] and is based on the polynomial approximation of both state and control. The state dynamics differential equation is then transformed into an equality constraint and the OCP is solved using NLP techniques.

An introductory survey of numerical methods for trajectory optimization may be found in Ref. [11], where the possibility to use heuristic methods (in particular genetic algorithms) is also taken into account. In Ref. [45] an interesting collocation methods using B-splines is reported. In Ref. [46] a description of the most recent numerical techniques to solve OCP is reported and typical aerospace test cases are solved. In the following, the pseudospectral approach will be described in further details as it will employed to compare and validate the IPSO results. A pseudospectral approach is a direct method that is able to estimate the costate and verify the necessary conditions given the particular choice of the polynomial approximation functions and the node distribution.

In this work, we are reporting a classification of OCP based on the identification of the histories to be transcribed. In general, conversion of OCPs into POPs is accomplished by replacing the control and/or state histories by control and/or state parameters and forming the histories by interpolation [10]. Following the analysis reported in Ref. [47], we will consider the formulation of an OCP in the state-control (SC) space, using the differential-inclusion (DI) formalism and the differential flatness (DF) description. The goal of this analysis is to make the reader

understand the reasoning that has lead to the development of IPSO, which is based on a DF implementation.

When the control (alone or along with the state) is approximated, the SC space formulation is employed. When the state is approximated, the DI formulation may be employed. Lastly, when the minimal set of independent functions is approximated, the DF formulation must be considered.

2.3 An example problem

The following example problem has been taken from Ref. [34, 48]. The system has two degrees of freedom, q_1 and q_2 (angles measuread in rad), and one control input, u (torque, measured in N-m), and is described by a 4th order differential equation. The OCP is minimizing

$$J = \int_{t_0}^{t_f} u^2(t)dt \tag{2.1}$$

subject to the state dynamics equations

$$I_1\ddot{q}_1 + mgl\sin q_1 + k(q_1 - q_2) = 0, \tag{2.2}$$

$$I_2\ddot{q}_2 - k(q_1 - q_2) = u, \tag{2.3}$$

where the reported quantities are, in MKS units,

$$\begin{aligned} I_1 = I_2 = 1.0\,\text{kg}\cdot\text{m}^2, \quad k = 1.0\,\text{N-m}, \\ g = 9.8\,\text{m}\cdot\text{s}^{-2}, \quad m = 0.01\,\text{kg}, \quad l = 0.5\,\text{m}. \end{aligned} \tag{2.4}$$

End-point constraint are imposed as

$$[q_1(t_0), q_2(t_0), \dot{q}_1(t_0), \dot{q}_2(t_0)] = [0.03\,\text{rad}, 0.01\,\text{rad}, 0.04\,\text{rad/s}, 0.05\,\text{rad/s}], \tag{2.5}$$

$$[q_1(t_f), q_2(t_f), \dot{q}_1(t_f), \dot{q}_2(t_f)] = [0.06\,\text{rad}, 0.02\,\text{rad}, 0.08\,\text{rad/s}, 0.02\,\text{rad/s}], \tag{2.6}$$

and the control constraint is

$$|u(t)| \leq 15\,\text{N-m} \quad \forall t \in [t_0, t_f]. \tag{2.7}$$

2.4 State-Control Problem ($\mathsf{P_{SC}}$)

Let us introduce the generic state function $\boldsymbol{x}(\cdot)$ with values $\boldsymbol{x}(t) \in \mathbb{X} \subset \mathbb{R}^{N_x}$, where N_x is the dimension of the state space, and the generic control function $\boldsymbol{u}(\cdot)$ with values $\boldsymbol{u}(t) \in \mathbb{U} \subset \mathbb{R}^{N_u}$, where N_u is the dimension of the control space. The

Bolza problem defined by means of the state-control space formulation, $\mathrm{P_{SC}}$, is an optimization problem usually defined as

$$\begin{gathered}
\text{Find } \boldsymbol{x}(t) : t \to \mathbb{X} \subset \mathbb{R}^{N_x}, \boldsymbol{u}(t) : t \to \mathbb{U} \in \mathbb{R}^{N_u}, \, t_f \in \mathbb{R} \\
\text{minimizing} \\
J[\boldsymbol{x}(t), \boldsymbol{u}(t), t_0, t_f] = E(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f) + \int_{t_0}^{t_f} F(\boldsymbol{x}(t), \boldsymbol{u}(t), t)\, dt \\
\text{subject to, } \forall t \in [t_0, t_f] \\
\text{Dynamics constraints:} \quad \dot{\boldsymbol{x}}(t) = \mathbf{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \\
\text{Boundary constraints:} \quad \mathbf{b}(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f) = \mathbf{0}, \\
\text{Path Constraints:} \quad \mathbf{p}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \leq \mathbf{0}.
\end{gathered} \tag{2.8}$$

When studying the existence or the uniqueness of the solution to the above problem, $\boldsymbol{x}(\cdot)$ is usually supposed to be an absolutely continuous function (i.e., $\boldsymbol{x}(\cdot) \in AC[t_0, t_f]$) or a Lipschitz-continuous function (i.e., $\boldsymbol{x}(\cdot) \in W^{1,\infty}[t_0, t_f]$).

From the theoretical point of view, when solving a fixed-time problem, the total number of unknowns is given by the N_x components of the state function plus the N_u components of the control functions. For a free-time problem one more value must be determined corresponding to the total time of application of the control.

APPLICATION Considering the example problem of Sec. 2.3, we can introduce x_i, $i = 1, ..., N_x$, with $N_x = 4$, given as

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}. \tag{2.9}$$

For this example $N_u = 1$. The OCP is formulated as

$$\begin{aligned}
\text{minimize} \quad & J = \int_{t_0}^{t_f} u^2(t) dt \\
\text{subject to,} \quad & \forall t \in [t_0, t_f] \\
\text{dynamics constraints:} \quad & \dot{x}_1(t) = x_3(t) \\
& \dot{x}_2(t) = x_4(t) \\
& \dot{x}_3(t) = -\frac{mgl \sin x_1(t) + k(x_1(t) - x_2(t))}{I_1} \\
& \dot{x}_4(t) = \frac{k(x_1(t) - x_2(t)) + u(t)}{I_2}
\end{aligned} \tag{2.10}$$

$$\begin{aligned} \text{boundary constraints:} \quad & [x_1, x_2, x_3, x_4]_{t_0} = [0.03, 0.01, 0.04, 0.05] \\ & [x_1, x_2, x_3, x_4]_{t_f} = [0.06, 0.02, 0.08, 0.02] \\ \text{control constraint:} \quad & |u(t)| \leq 15 \end{aligned}$$

From the numerical point of view, the number of unknowns depends on the parametrization used for solving the optimization problem. For instance, a collocation method as well as a pseudospectral approach would transcribe the optimization problem approximating both $\boldsymbol{x}$ and $\boldsymbol{u}$ with polynomial expressions [40]. As a consequence, for the proposed problem we would have $(N_x + N_u)N_{\mathcal{P}} = 5N_{\mathcal{P}}$ optimization parameters for a fixed-time problem and $5N_{\mathcal{P}} + 1$ optimization parameters for a free-time problem. With this numerical approach, the differential equations related to the dynamics constraints in Eq. (2.10) are treated as equality constraints.

Another approach that can be used to reduce the number of the optimization parameters is given by approximating the control and integrating the state-dynamics equation to obtain the state. In this case, the number of optimization parameters is reduced to $N_u N_{\mathcal{P}} = N_{\mathcal{P}}$ for a fixed-time problem and $N_{\mathcal{P}} + 1$ for a free-time problem. With this approach, the differential equations related to the dynamics constraints in Eq. (2.10) are treated as ODEs requiring a numerical solver (e.g., a Runge-Kutta scheme). Spiller *et al.*[1] have considered this approach, referred to as direct dynamics method, to solve the same example problem considered here.

2.4.1 Direct dynamics method

When only the control is transcribed into a polynomial approximation, we deal with what is known as a direct dynamic approach. In this case, the control $\boldsymbol{u}$ is substituted by its numerical approximation $\boldsymbol{u}^N$ which is given by

$$\boldsymbol{u}^N = \sum_{i=0}^{N_{\mathcal{P}}-1} \tilde{u}_i \phi_i(t) \tag{2.11}$$

where $\phi_i(t)$ is the ith polynomial basis weighted by the coefficient $\tilde{u}_i$. In this approach, the state is obtained integrating the dynamics dictated by the approximated control. Usually, $N_{\mathcal{P}}$ coefficients are introduced, and the time is discretized into $N_T + 1$ points, i.e. $t = [t_0 < t_1 < ... < t_{N_T}]$. Different numerical approaches can be developed changing the polynomial bases, which can be Lagrange polynomial, Chebyshev polynomial or B-Splines. In Ref. [1], the author of this thesis presented a method based on a third-degree spline approximation, which will be taken into account in Sec. 5.3. In that paper, the author used the direct method to solve a constrained attitude reorientation problem.

The set of parameters involved in the optimization process are $[\tilde{u}_0, ..., \tilde{u}_{N_p-1}]$ for a fixed time problem and $[t_f, \tilde{u}_0, ..., \tilde{u}_{N_p-1}]$ for a free-time problem. The OCP is transcribed such that an approximated cost function is introduced, J^N, given as

$$J^N = E(\boldsymbol{x}^N(t_0), \boldsymbol{x}^N(t_f), t_0, t_f) + I^N(F) \tag{2.12}$$

where $\boldsymbol{x}^N$ is obtained via numerical integration and $I^N(F)$ is the numerical approximation of the running cost function integration. For example, if a trapezoidal method is used, this term is given as

$$J^N = \frac{t_f - t_0}{2N_T} \sum_{j=1}^{N_T} \left(F_{j-1}^N + F_j^N \right), \tag{2.13}$$

where

$$F_j^N = F(\boldsymbol{x}^N(t_j), \boldsymbol{u}^N(t_j), t_j). \tag{2.14}$$

The problems is then transcribed into a constrained NLP problem that can be solved with one of the numerical techniques described in Sec. 1.3.2.

2.4.2 Collocation methods and pseudospectral approaches

Another class of approaches based on the state-control formulation is the collocation method. In this case, both the state and the control are discretized, i.e.

$$\boldsymbol{x}^N = \sum_{i=0}^{N_\mathcal{P}-1} \tilde{x}_i \phi_i(t) \tag{2.15}$$

$$\boldsymbol{u}^N = \sum_{i=0}^{N_\mathcal{P}-1} \tilde{u}_i \phi_i(t) \tag{2.16}$$

The idea is to represent states and controls by polynomials that can be easily integrated and differentiated. The following principles are applied:

1. The states and controls at nodes (depending on the discretization of the time domain) are taken as free parameters.

2. Between nodes, states and controls are represented by polynomials. In the local approach, piecewise polynomials are exploited, i.e. different polynomial are defined for each interval; in the global approach, a unique polynomial is used.

3. State rates at the nodes are calculated by the dynamic equations.

4. Implicit integration is performed to enforce the dynamic equation at the segment's center.

5. A constrained NLP problem is solved with one of the numerical techniques described in Sec. 1.3.2.

In Ref. [44], a local method employing cubic splines has been presented. In that paper, the algorithm for the direct numerical solution of an optimal control problem was based on cubic polynomials to represent state variables. The control variables were linearly interpolated and collocation was used to satisfy the differential equations.

In recent years pseudospectral methods have increased in popularity. An introduction to this family of methods is reported in Ref. [49]. A pseudospectral method is a global form of orthogonal collocation (see Ref. [40, 50]), i.e., in a pseudospectral method the state and the control are approximated using a global polynomial and collocation is performed at chosen points. The basis functions are typically Chebyshev or Lagrange polynomials. Pseudospectral methods were developed originally to solve problems in computational fluid dynamics. The rationale for using a global polynomial with orthogonally collocated points is that the approximation will converge spectrally (i.e., at an exponential rate) as a function of the number of collocation points (see Ref. [51]).

In Ref. [52], moreover, a comparison between local and global pseudospectral methods has been reported. The characteristic of this class of methods is that they can provide an estimation of the costate, as reported in Ref. [40, 53, 54, 55, 56], through the Covector Mapping Principle.

Using the pseudospectral approach, some of the currently most-used commercial software for optimal control have been developed, i.e. DIDO (based on the theory described in Ref. [27]) and GPOPS (described in Ref. [57]).

2.5 Differential Inclusion Problem (P_{DI})

An approach often used in engineering [58, 59, 60] and mathematics [36, 37, 61] to solve the previous problem is based on the formulation of the Differential Inclusion Problem P_{DI}. Such DI problems are natural generalizations of free/fixed time problems in both the calculus of variations and optimal control[62]. Moreover, DI models allow one to consider closed-loop control systems with control regions where $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$.

With this formulation, the control $\boldsymbol{u}(t)$ does not appear explicitly in the cost function, i.e. the OCP is written as

$$\begin{aligned}
&\text{Find } \boldsymbol{x}(t) : t \to \mathbb{X} \subset \mathbb{R}^{N_x},\ t_f \in \mathbb{R} \\
&\text{minimizing} \\
&J[\boldsymbol{x}(\cdot), t_f] = E(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} F\left(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t\right) dt \\
&\text{subject to, } \forall t \in [t_0, t_f] \\
&\text{Dynamics constraints:} \quad \dot{\boldsymbol{x}}(t) \in \boldsymbol{\mathcal{F}}(\boldsymbol{x}(t)), \\
&\text{Boundary constraints:} \quad \mathbf{b}(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_0, t_f) = \mathbf{0}, \\
&\text{Path Constraints:} \quad \mathbf{p}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t)) \leq \mathbf{0}.
\end{aligned} \tag{2.17}$$

In this case, the optimal control problem is defined over all arcs $\boldsymbol{x}(\cdot)$ satisfying the differential inclusion

$$\dot{\boldsymbol{x}}(t) \in \boldsymbol{\mathcal{F}}(\boldsymbol{x}(t)) \quad \text{a.e. on } [t_0, t_f] \tag{2.18}$$

where

$$\boldsymbol{\mathcal{F}}(\boldsymbol{x}(t)) = \{\boldsymbol{v} \in \boldsymbol{\mathcal{X}} : \boldsymbol{v} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)),\ \boldsymbol{u}(t) \in \boldsymbol{\mathcal{U}}(t)\} \tag{2.19}$$

From the numerical point of view, this approach is advantageous if Eq. (2.18) admits a control of the form

$$\boldsymbol{u}(t) = \boldsymbol{g}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t)). \tag{2.20}$$

In this case, the control set is given as $\boldsymbol{\mathcal{U}}(\boldsymbol{x}(t), t)$ and Eq. (2.19) may be rewritten as

$$\boldsymbol{\mathcal{F}}(\boldsymbol{x}(t)) = \{\boldsymbol{v} \in \boldsymbol{\mathcal{X}} : \boldsymbol{v} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)),\ \boldsymbol{u}(t) \in \boldsymbol{\mathcal{U}}(\boldsymbol{x}(t), t)\} \tag{2.21}$$

APPLICATION For the example problem proposed in Sec. 2.3, a solution can be obtained by means of the DI approach approximating only the state and substituting $\boldsymbol{u}$ with

$$u = I_2 \dot{x_4} - k(x_1 - x_2) \tag{2.22}$$

The end-point constraints are expressed in the same way as in Sec. 2.4, but the remaining of the OCP is modified as

$$\begin{aligned}
\text{minimize} \quad & J = \int_{t_0}^{t_f} (I_2 \dot{x_4} - k(x_1 - x_2))^2(t) dt \\
\text{subject to,} \quad & \forall t \in [t_0, t_f] \\
\text{dynamics constraints:} \quad & \dot{x}_1 = x_3 \\
& \dot{x}_2 = x_4 \\
& \dot{x}_3 = -\frac{mgl \sin x_1 + k(x_1 - x_2)}{I_1}
\end{aligned} \tag{2.23}$$

$$
\begin{aligned}
\text{boundary constraints:} \quad & [x_1, x_2, x_3, x_4]_{t_0} = [0.03, 0.01, 0.04, 0.05] \\
& [x_1, x_2, x_3, x_4]_{t_f} = [0.06, 0.02, 0.08, 0.02] \\
\text{control constraint:} \quad & |I_2 \dot{x_4} - k(x_1 - x_2)| \leq 15
\end{aligned}
$$

As can be seen, DI formulation allows one to reduce the number of unknown quantities to be determined with respect to the state-control space formulation. In fact, the control $\boldsymbol{u}(t)$ is no longer an independent function but is obtained directly from the dynamics equation.

At this point, one can choose to approximate all the N_x components of the state function and treat the dynamics constraint equations in Eq. (2.23) as an equality constraint. Hence, when solving a fixed-time problem, the total number of unknowns is equal to $N_x N_{\mathcal{P}} = 4N_{\mathcal{P}}$ whereas, for a free-time problem, it is $4N_{\mathcal{P}} + 1$.

2.5.1 Differential inclusion transcription

In this case, only the state is transcribed, i.e.

$$
\boldsymbol{x}^N = \sum_{i=0}^{N_{\mathcal{P}}-1} \tilde{x}_i \phi_i(t). \tag{2.24}
$$

The control is obtained from the state differential equation and does not need to be approximated. On the contrary, the control does exactly satisfy the state differential equation. The remaining differential equation concerning only the state are treated as equality constraints.

The advantage of a differential inclusion approach over the common collocation approach is not completely assessed. In Ref. [63], the authors states that the DI approach leads to some computational advantages but the difficulties lie in the formulation of an OCP in differential inclusion form. In Ref. [64, 33], the author state that the differential inclusion formulation usually requires more iterations to converge to results with the same precision of a collocation approach. However, these conclusions highly depends on the numerical techniques employed for the polynomial approximation, the NLP used to solve the problem and other technical details defining the numerical procedure.

2.6 Differential Flatness Problem (P_{DF})

Let us suppose that a flat output $\boldsymbol{y}(t)$, with the same dimension as the control $\boldsymbol{u}(t)$, can be defined as (see Ref. [32]),

$$\boldsymbol{y} = \boldsymbol{c}\left(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, ..., \boldsymbol{u}^{(\alpha)}\right), \quad \boldsymbol{y} \in \mathbb{R}^{N_u}, \tag{2.25}$$

where

$$\boldsymbol{u}^{(\alpha)} = \frac{d^{\alpha}\boldsymbol{u}}{dt^{\alpha}} \tag{2.26}$$

with $\alpha \in \mathbb{N}$. If such a flat output exists for the dynamical system taken into account, then the state and the control can be written as

$$\boldsymbol{x} = \boldsymbol{a}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, ..., \boldsymbol{y}^{(\beta)}\right), \quad \boldsymbol{u} = \boldsymbol{b}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, ..., \boldsymbol{y}^{(\beta+1)}\right), \tag{2.27}$$

where the value of $\beta \in \mathbb{N}$ depends on the problem. Let us introduce a particular notation, concerning the set of the time derivatives of $\boldsymbol{y}(t)$ from the first derivative to the ν-th derivative, given as

$$\{\boldsymbol{y}^{(\nu)}(t)\} = \{\dot{\boldsymbol{y}}(t), \ddot{\boldsymbol{y}}(t), ..., \boldsymbol{y}^{(\nu)}(t)\}. \tag{2.28}$$

Consequently, the Bolza problem can be stated as a function of the flat output as

$$\begin{gathered}
\text{Find } \boldsymbol{y}(t) : t \to \mathbb{Y} \subset \mathbb{R}^{N_u},\ t_f \in \mathbb{R} \\
\text{minimizing} \\
J[\boldsymbol{y}(\cdot), t_f] = E(\boldsymbol{y}(t_0), \{\boldsymbol{y}^{(\beta)}(t_0)\}, \boldsymbol{y}(t_f), \{\boldsymbol{y}^{(\beta)}(t_f)\}, t_f) \\
+ \int_{t_0}^{t_f} F\left(\boldsymbol{y}(t), \{\boldsymbol{y}^{(\beta)}(t)\}, t\right) dt \\
\text{subject to, } \forall t \in [t_0, t_f] \\
\text{Boundary constraints:} \quad \mathbf{b}(\boldsymbol{y}(t_0), \{\boldsymbol{y}^{(\beta)}(t_0)\}, \boldsymbol{y}(t_f), \{\boldsymbol{y}^{(\beta)}(t_f)\}, t_f) = \mathbf{0}, \\
\text{Path Constraints:} \quad \mathbf{p}(\boldsymbol{y}(t), \{\boldsymbol{y}^{(\beta+1)}(t)\}, t) \leq \mathbf{0}.
\end{gathered} \tag{2.29}$$

Using the differentially flat approach, the dynamical system is expressed as a static system [47, 65]. In other words, there are no more differential constraints and both the state and the control are expressed as an explicit function of the flat output and its time derivatives.

APPLICATION For the example problem proposed in Sec. 2.3, we can choose as flat output $y = x_1$. With few simple mathematical manipulations, it can be shown that

$$x_1 = y, \tag{2.30a}$$

$$x_2 = \frac{I_1\ddot{y} + mgl\sin y}{k_1} + y, \tag{2.30b}$$

$$x_3 = \dot{y}, \tag{2.30c}$$

$$x_4 = \frac{I_1 y^{(3)} + mgl\dot{y}\cos y}{k_1} + \dot{y}, \tag{2.30d}$$

$$u = \frac{y^{(4)}(\beta_2\cos y + \beta_3)\ddot{y} - (\beta_4\dot{y}^2 + \beta_5)\sin y}{\beta_1}, \tag{2.30e}$$

where

$$\beta_1 = \frac{k}{I_1 I_2}, \quad \beta_2 = -\frac{mgl}{I_1}, \quad \beta_3 = -\frac{k(I_1 + I_2)}{I_1 I_2} \tag{2.31}$$

$$\beta_4 = -\beta_2, \quad \beta_5 = -\frac{mkgl}{I_1 I_2}. \tag{2.32}$$

As a consequence of this choice, the OCP is formulated as

$$\begin{aligned}
&\text{minimize} \quad J = \frac{1}{\beta_1^2}\int_{t_0}^{t_f}\Big(y^{(4)}(\beta_2\cos y + \beta_3)\ddot{y} - (\beta_4\dot{y}^2 + \beta_5)\sin y\Big)^2 dt \\
&\text{subject to,} \quad \forall t \in [t_0, t_f] \\
&\text{boundary constraints:} \quad \begin{bmatrix} y \\ \dfrac{I_1\ddot{y} + mgl\sin y}{k_1} + y \\ \dot{y} \\ \dfrac{I_1 y^{(3)} + mgl\dot{y}\cos y}{k_1} + \dot{y} \end{bmatrix}_{t_0} = \begin{bmatrix} 0.03 \\ 0.01 \\ 0.04 \\ 0.05 \end{bmatrix} \\
&\qquad\qquad\qquad\qquad \begin{bmatrix} y \\ \dfrac{I_1\ddot{y} + mgl\sin y}{k_1} + y \\ \dot{y} \\ \dfrac{I_1 y^{(3)} + mgl\dot{y}\cos y}{k_1} + \dot{y} \end{bmatrix}_{t_f} = \begin{bmatrix} 0.06 \\ 0.02 \\ 0.08 \\ 0.02 \end{bmatrix} \\
&\text{control constraint:} \quad \left|\frac{y^{(4)}(\beta_2\cos y + \beta_3)\ddot{y} - (\beta_4\dot{y}^2 + \beta_5)\sin y}{\beta_1}\right| \leq 15
\end{aligned} \tag{2.33}$$

This approach leads to the minimum number of unknown parameters without requiring a numerical integration ($N_{\mathcal{P}}$ for a fixed-time problem, $N_{\mathcal{P}} + 1$ for a free time problem). Note that the time derivatives of y, i.e. $\dot{y}$, $\ddot{y}$, $y^{(3)}$ and $y^{(4)}$, are not independent functions as they are derived analytically from y once the specific polynomial approximation is imposed.

The issue often related to this approach is that it can add some difficulties to the cost function after the transcription has been carried out.

2.6.1 Differential flatness transcription

In this case, only the flat output is transcribed, i.e.

$$\boldsymbol{y}^N = \sum_{i=0}^{N_{\mathcal{P}}-1} \tilde{y}_i \phi_i(t). \tag{2.34}$$

The control and the state are obtained as closed form solution once the values of the flat output are given. As for the differential inclusion, the control does exactly satisfy the state differential equation. The problem is no more subject to dynamical constraints.

Substituting Eq. (2.34) into Eq. (2.33), we can see that the cost function becomes a nonconvex function of the optimization parameters. It is noteworthy that, for both P_{SC} and P_{DI}, the transcribed cost function was a convex function of the optimization parameters. The nonconvexity issue related to the DF formulation can be overcome with global metaheuristic optimization techniques.

2.7 Endnotes

This chapter has focused on the key-role played by the problem formulation when an OCP is to be solved. As the core of the thesis is the development of the Inverse-dynamics Particle Swarm Optimization, the philosophy of this chapter has been to guide the reader into the process that allows one to formulate an optimal control problem with the differential flatness formalism. However, it is a firm opinion of the author that trying to identify a unique, global approach to solve all the optimal control problems is not the best way to search for solutions. Every single optimization problem has its own characteristics, and different formulations of the transcribed parameter optimization can lead to different solutions. To conclude, the final remark of this chapter is that the formulation of the optimal control problem quite almost defines the solution to the problem itself.

Part II

Solution of Optimal Control Problem via Swarm Intelligence

3 Particle Swarm Optimization

> *Learning would be exceedingly laborious, not to mention hazardous, if people had to rely solely on the effects of their own actions to inform them what to do.*
>
> **— Albert Bandura**
> (American psychologist)

Abstract

In this chapter the Particle Swarm Optimization is described. The origin of the method, its connection with sociological investigations and the main mathematical features of the algorithms are presented. This metaheuristic method is employed in the Inverse-dynamics Particle Swarm Optimization since it mixes a very easy and straightforward numerical implementation with the ability to search for the global optimum in hard optimization problem affected by nonlinearities and nonconvexity issues.

Nomenclature

$N_{\mathcal{S}}$	=	Number of particles in the swarm	$r_{(\cdot)}$	=	Random number
$\mathbf{x}$	=	Particle in the swarm	K	=	Coefficient of the PSO-parameters
$\mathbf{v}$	=	Velocity of the particles	$\boldsymbol{l}_{best}$	=	Local best particle
J	=	Performance index	$\boldsymbol{p}_{best}$	=	Personal best particle
J_l	=	Local best performance index	$\boldsymbol{g}_{best}$	=	Global best particle
J_p	=	Personal best performance index	$\mathcal{N}$	=	Local search neighborhood
J_g	=	Global best performance index	L_R	=	Local search radius
c_l	=	Local parameter	$(\cdot)_i$	=	Feature of the ith particle
c_p	=	Personal parameter	$(\cdot)^{(k)}$	=	Value at the kth iteration
c_g	=	Global parameter	$(\cdot)^0$	=	Value at the first iteration
w	=	Inertia weight	$(\cdot)^f$	=	Value at the final iteration
$\mathbf{x}^*$	=	Minimizing particle	J^*	=	Minimum performance index
$\hat{k}$	=	Index for $\boldsymbol{g}_{best}$ updating	ε	=	Convergence threshold
N_{eval}	=	Number of evaluation of J	t_{CPU}	=	Computational time

3.1 Introduction

The Particle Swarm Optimization (PSO) has been firstly introduced in 1995 by J. Kennedy and R. C. Eberhart in Ref. [66]. After that, several papers and books have been published dealing with PSO. Among them, Ref. [67] explains the relationships of the optimization technique with psychological and sociological research fields, while Ref. [68] focuses on mathematical and implementation details.

Among the several metaheuristic algorithms developed until present days, PSO is one of the most used technique, as demonstrated by the several works in literature relying on this method (e.g., see Ref. [67, 68, 69]). PSO belongs to the family of swarm intelligence algorithms, it is based on very simple formulas and it requires a straightforward implementation. The most interesting feature of PSO is its relationships with nature and sociological issues. The algorithm was born to simulate the social behavior of bird flocking or fish schooling. The contributions in the displacement of the particle are related to speculations related to the interaction of human beings. Knowledge and welfare related to a specific person can improve when the individual is included in a social contest where interaction and communication play a fundamental role. This extremely interesting investigations can be found in Ref. [67], a book written by one of the creators of the PSO, James Kennedy, which actually is a social psychologist.

PSO is a metaheuristic algorithm with enhanced ability to perform global optimization [67]. The method exploits a fixed-size population of *particles*, i.e. candidate solutions containing the optimization parameters. The *swarm* includes $N_{\mathcal{S}}$ particles (typically $N_{\mathcal{S}} \in \{30, ..., 50\}$) which move inside the Search Space (SS) modifying their *position* (i.e., the values of the parameters associated with it) by means of an appropriate perturbation called *velocity*. The velocity term at the k^{th} iteration is usually given by several contributions evaluated on the basis of the *Performance Index* $J_i^{(k)}$ of every i^{th} particle, a measure of the particle optimality which takes into account the goal of the optimization and the imposed constraints. The objective of the PSO algorithm is the minimization of the performance index.

The chapter is organized as follows. In Sec. 3.2 the original global version of the PSO is described. In Sec. 3.3 the socio-cognitive background is given, stating the relationship with sociological investigations that characterizes the optimizer, while in Sec. 3.4 the origin of the terms “swarm” and “particle” is explained. In Sec. 3.5 the local version of the PSO is presented, which is extremely useful in cases where several local minima exist. The exploration and exploitation abilities of the PSO are investigated in Sec. 3.6, and the unified paradigm is presented in Sec. 3.7. Some notes on the PSO convergence properties are given in Sec. 3.8 and concluding remarks are reported in Sec. 3.9.

3.2 Global version

Although the original intent was to graphically simulate the graceful but unpredictable choreography of a bird flock (see Ref. [66]), the first version of the future PSO algorithm was quite soon recognized as a good optimization method.

The very first model of PSO is the so called "global version". Let us introduce the m-dimensional particle $\boldsymbol{x}_i \in \mathbb{R}^m$, $i = 1, ..., N_{\mathcal{S}}$, where $N_{\mathcal{S}}$ is the number of particles within the swarm. The particle contains the optimization variables of the problem. The PSO is based on the definition of a perturbation term called "velocity" which allows the particle to perform a "displacement" to explore the search space.

Let us firstly consider the primitive formulation reported in Ref. [66]. With reference to Fig. 3.1, the generic i^{th} particle at the iteration k, $\mathbf{x}_i^k$, moves from the iteration k to the iteration $k+1$ according to the rule

$$\mathbf{x}_i^{(k+1)} = \mathbf{x}_i^{(k)} + \mathbf{v}_i^{(k+1)}, \tag{3.1}$$

where $\mathbf{v}_i^{(k+1)}$ is the velocity required to pass from the iteration k to the next one, given by

$$\mathbf{v}_i^{(k+1)} = \mathbf{v}_i^{(k)} + r_1 c_p \left(\boldsymbol{p}_{best,i}^{(k)} - \mathbf{x}_i^{(k)}\right) + r_2 c_g \left(\boldsymbol{g}_{best}^{(k)} - \mathbf{x}_i^{(k)}\right) \tag{3.2}$$

where r_1 and r_2 are random number uniformly distributes in $[0, 1]$ and c_p and c_g are parameters set equal to 2 in Ref. [66]. Note that the term $\mathbf{v}_i$ is not a velocity in the traditional physical sense, i.e. $\mathbf{v}_i \neq d\mathbf{x}_i/dt$, but the velocity is the rate at which the position per generation changes.

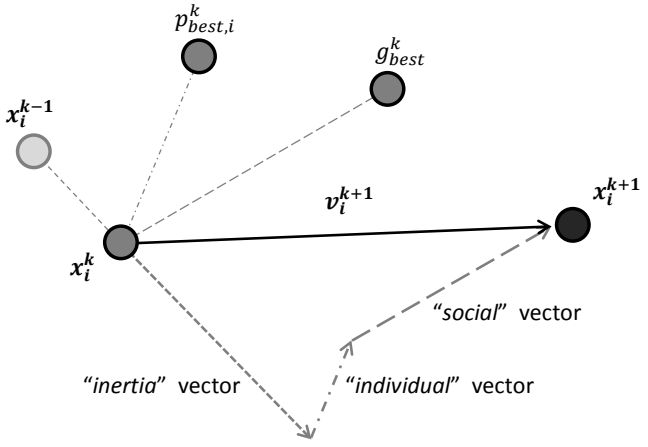


Fig. 3.1: Displacement of a particle in the global paradigm.

As already stated in the introduction of this chapter, the velocity term at the k^{th} iteration is given by several contributions evaluated on the basis of the *Performance Index* $J_i^{(k)}$ of every i^{th} particle, a measure of the particle optimality which takes into account the goal of the optimization and the imposed constraints. The performance index is a function depending on the particle position, i.e.

$$J_i^{(k)} = f(\mathbf{x}_i^{(k)}), \tag{3.3}$$

where the function f is defined by the optimization problem. The three contributions in Eq. (3.2) are defined as:

- *Inertia* vector $\mathbf{v}_i^{(k)}$, pointing from $\mathbf{x}_i^{(k-1)}$ to $\mathbf{x}_i^{(k)}$.

- *Individual* vector $\boldsymbol{p}_{best,i}^{(k)} - \mathbf{x}_i^{(k)}$, pointing to the *personal best* $\boldsymbol{p}_{best,i}^{(k)}$ given by $J_{p,i}^{(k)} = \min\limits_{1<j<k} J_i^{(j)}$. In particular,

$$\text{if} \quad \min_{1<j<k} J_i^{(j)} < J_{p,i}^{(k)} \Rightarrow J_{p,i}^{(k)} = \min_{1<j<k} J_i^{(j)} \Rightarrow \boldsymbol{p}_{best,i}^{(k)} = \boldsymbol{x}_i^{(k)} \tag{3.4}$$

- *Global search* vector $\boldsymbol{g}_{best,i}^{(k)} - \mathbf{x}_i^{(k)}$, pointing to the *global best* $\boldsymbol{g}_{best}^{(k)}$ given by $J_g^{(k)} = \min\limits_{1\leq j\leq N_{\mathcal{S}}} J_{p,i}^{(k)}$. In particular,

$$\text{if} \quad \min_{1\leq j\leq N_{\mathcal{S}}} J_{p,i}^{(k)} < J_g^{(k)} \Rightarrow J_g^{(k)} = \min_{1\leq j\leq N_{\mathcal{S}}} J_{p,i}^{(k)} \Rightarrow \boldsymbol{g}_{best}^{(k)} = \boldsymbol{p}_{best,i}^{(k)} \tag{3.5}$$

The variables $\boldsymbol{p}_{best,i}$ and $\boldsymbol{g}_{best}$ have been introduced for different reasons. Conceptually, $\boldsymbol{p}_{best,i}$ resembles autobiographical memory, as each i-th individual remembers its own experience (though only one fact about it), and the velocity adjustment associated with $\boldsymbol{p}_{best,i}$ can be called "simple nostalgia" in that the individual tends to return to the place that most satisfied it in the past. On the other hand, $\boldsymbol{g}_{best}$ is conceptually similar to publicized knowledge, or a group norm or standard, which individuals seek to attain. In the simulations, a high value of the $\boldsymbol{p}_{best,i}$-increment relative to the $\boldsymbol{g}_{best}$-increment results in excessive wandering of isolated individuals through the problem space, while the reverse (relatively high $\boldsymbol{g}_{best}$-increment) results in the flock rushing prematurely toward local minima. Approximately equal values of the two increments seem to result in the most effective search of the problem domain.

In Ref. [70] a refined version of Eq. (3.2) has been introduced, which is

$$\mathbf{v}_i^{(k+1)} = w\mathbf{v}_i^{(k)} + r_1 c_p \left(\boldsymbol{p}_{best,i}^{(k)} - \mathbf{x}_i^{(k)}\right) + r_2 c_g \left(\boldsymbol{g}_{best}^{(k)} - \mathbf{x}_i^{(k)}\right). \tag{3.6}$$

Here, a fixed *inertia weight* has been introduced to properly weight the previous particle velocity with respect to the other two contributions. In Ref. [70], the authors suggest to use a value of w in $[0.9, 1.2]$. However, a linearly decreasing w is also suggested, such that

$$\mathbf{v}_i^{(k+1)} = w^{(k)}\mathbf{v}_i^{(k)} + r_1 c_p \left(\boldsymbol{p}_{best,i}^{(k)} - \mathbf{x}_i^{(k)}\right) + r_2 c_g \left(\boldsymbol{g}_{best}^{(k)} - \mathbf{x}_i^{(k)}\right). \tag{3.7}$$

and

$$w^{(k)} = \begin{cases} w^0 - (w^0 - w^f)\dfrac{k-1}{K} & \text{if } k \leq K\,, \\ w^f & \text{if } k > K\,. \end{cases} \tag{3.8}$$

where K is a user-defined parameter, w_0 and w_f are the initial and final values, respectively. In Ref. [70], suggested values for the parameters in Eq. (3.8) are $K = 4000$, $w^0 = 1.4$ and $w^f = 0$, while in Ref. [71] $w^0 = 0.9$ and $w^f = 0.4$, with $K \in \{1000, 1500, 2000\}$.

The particles move only within the SS since the velocity and the displacement are imposed to lie inside the iper-parallelepids with lower limits $\mathbf{v}_{min}$ and $\mathbf{x}_{min}$ and upper limit $\mathbf{v}_{max}$ and $\mathbf{x}_{max}$, respectively. Most usually, $\mathbf{v}_{min} = -\mathbf{v}_{max}$ whereas $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ depend on the optimization problem. As a consequence, the following inequality constraints must be satisfied:

$$\mathbf{v}_{min} \leq \mathbf{v} \leq \mathbf{v}_{max}\,, \quad \mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max} \tag{3.9}$$

The maximum and minimum values of the velocity and the displacement are defined by the user. Usually, the maximum value of the velocity is set at about $10 - 20\%$ of the dynamic range of the variable (which is given by $\mathbf{x}_{max} - \mathbf{x}_{min}$).

3.3 Socio-cognitive background

In their first pioneering work, Ref. [66], J. Kennedy and R. C. Eberhart introduced the relationships between particle swarm optimization and both artificial life and genetic algorithms.

As already stated, the PSO is a method for optimization of continuous nonlinear functions that has been discovered through simulation of a simplified social model. PSO has roots in two main component methodologies. Perhaps more obvious are its ties to artificial life (A-life) in general, and to bird flocking, fish schooling, and swarming theory in particular. It is also related, however, to evolutionary computation, and has ties to both genetic algorithms and evolutionary programming.

PSO comprises a very simple concept, and paradigms can be implemented in a few lines of computer code. It requires only primitive mathematical operators, and is computationally inexpensive in terms of both memory requirements and speed. Early testing has found the implementation to be effective with several kinds of problems (see Ref. [66]).

The origins of the PSO lie in the study of the processes that underlie the unpredictable group dynamics of bird social behavior, i.e. the underlying rules that enabled large numbers of birds to flock synchronously, often changing direction suddenly, scattering and regrouping, etc. It was thought that manipulation of inter-individual distances was of primary importance; that is, the synchrony of flocking behavior was thought to be a function of birds' efforts to maintain an optimum distance between themselves and their neighbors. A fundamental hypothesis to the development of particle swarm optimization was the belief that social sharing of information among conspeciates offers an evolutionary advantage.

Besides, another motive for developing the simulation was to model human social behavior, which is of course not identical to fish schooling or bird flocking. One important difference is its abstractness. Birds and fish adjust their physical movement to avoid predators, seek food and mates, optimize environmental parameters such as temperature, etc. Humans adjust not only physical movement but cognitive or experiential variables as well. We do not usually walk in step and turn in unison; rather, we tend to adjust our beliefs and attitudes to conform with those to our social peers.

Particle swarm optimization is an extremely simple algorithm that seems to be effective for optimizing a wide range of functions. We view it as a mid-level form of A-life or biologically derived algorithm, occupying the space in nature between evolutionary search, which requires eons, and neural processing, which occurs on the order of milliseconds. Social optimization occurs in the time frame of ordinary experience - in fact, it is ordinary experience. In addition to its ties with A-life, particle swarm optimization has obvious ties with evolutionary computation. Conceptually, it seems to lie somewhere between genetic algorithms and evolutionary programming. It is highly dependent on stochastic processes, like evolutionary programming. The adjustment toward $\boldsymbol{p}_{best,i}$ and $\boldsymbol{g}_{best}$ by the PSO is conceptually similar to the crossover operation utilized by genetic algorithms. It uses the concept fitness, as do all evolutionary computation paradigms.

3.4 The origin of the terminology

The term swarm has a basis in the literature. In particular, the authors of PSO use the term in accordance with a paper by Millonas [72],who developed his swarm models for applications in artificial life, and articulated five basic principles of swarm intelligence. They are reported here in the following list:

P1 *Proximity principle*: the population should be able to carry out simple space and time computations.

P2 *Quality principle*: the population should be able to respond to quality factors in the environment.

P3 *Principle of diverse response*: the population should not commit its activities along excessively narrow channels.

P4 *Principle of stability*: the population should not change its mode of behavior every time the environment changes.

P5 *Principle of adaptability*: the population must be able to change behavior mode when it's worth the computational price.

Note that principles **P4** and **P5** are the opposite sides of the same coin. The particle swarm optimization concept and paradigm presented in Sec. 3.2 seem to adhere to all five principles. Basic to the paradigm are m-dimensional space calculations carried out over a series of time steps (**P1**). The population is responding to the quality factors $\boldsymbol{p}_{best,i}$ and $\boldsymbol{g}_{best}$ (**P2**). The allocation of responses between $\boldsymbol{p}_{best,i}$ and $\boldsymbol{g}_{best}$ ensures a diversity of response (**P3**). The population changes its state (mode of behavior) only when $\boldsymbol{g}_{best}$ changes, thus adhering to the principle of stability (**P4**). The population is adaptive because it does change when $\boldsymbol{g}_{best}$ changes (**P5**).

The term particle was selected as a compromise. While it could be argued that the population members are mass-less and volume-less, and thus could be called "points", it is felt that velocities and accelerations are more appropriately applied to particles, even if each is defined to have arbitrarily small mass and volume. Thus the label the authors have chosen to represent the optimization concept is particle swarm.

3.5 Local version

In Ref. [73] a locally-oriented paradigm of the PSO has been introduced. With reference to Fig. 3.2, the velocity perturbation is evaluated as

$$\mathbf{v}_i^{(k+1)} = w^{(k)}\mathbf{v}_i^{(k)} + r_1 c_p \left(\boldsymbol{p}_{best,i}^{(k)} - \mathbf{x}_i^{(k)}\right) + r_2 c_g \left(\boldsymbol{l}_{best,i}^{(k)} - \mathbf{x}_i^{(k)}\right). \tag{3.10}$$

where the local search vector $\boldsymbol{l}_{best,i}^{(k)} - \mathbf{x}_i^{(k)}$ is defined, pointing to the *local best* $\boldsymbol{l}_{best}^{(k)}$ given by $J_{l,i}^{(k)} = \min_{j\in\mathcal{N}_i} J_{p,i}^{(k)}$. The neighborhood $\mathcal{N}_i$ is a set containing the particles from $i - L_R$ to $i + L_R$, where L_R is a user-defined parameter ($\mathcal{N}_i$ is properly defined to consider the boundary cases where $i - L_R \leq 0$ and $i + L_R > N_S$). In particular,

$$if \quad \min_{j\in\mathcal{N}_i} J_{p,i}^{(k)} < J_{l,i}^{(k)} \Rightarrow J_{l,i}^{(k)} = \min_{j\in\mathcal{N}_i} J_{p,i}^{(k)} \Rightarrow \boldsymbol{l}_{best,i}^{(k)} = \boldsymbol{p}_{best,i}^{(k)} \tag{3.11}$$

The advantage of the local paradigm is that, usually, a number of groups of particles spontaneously separate and explore different regions of the SS. It is thus a more flexible approach to information processing than the $\boldsymbol{g}_{best}$ model. However, generally this is implementation is slower than the global paradigm to converge toward user-defined exit-tolerances.

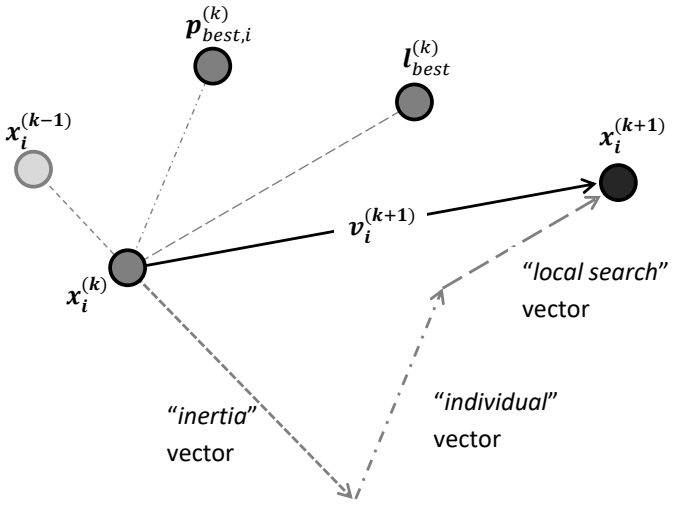


Fig. 3.2: Displacement of a PSO particle in the local paradigm.

3.6 Exploration and exploitation

As already discussed in Sec. 1.3.3, exploration and exploitation are two key aspects of a metaheuristic technique.

The exploration ability of the PSO may be enhanced in several ways, e.g. using the local paradigm or setting high values of $\mathbf{v}_{max}$ and w. The exploitation ability, instead, is increased with the global paradigm or setting low values of $\mathbf{v}_{max}$ and w. An inspection of the contribution of the several factors is reported in Ref. [74].

The best compromise is usually given by trying to enhance the exploration ability at the beginning of the optimization process. Toward the end of the search, the exploration ability should be improved in order to refine the final solution. The transition from exploration to exploitation of the SS can be obtained defining adaptive coefficients weighting the different velocity contributions. The following unified version of the PSO is the approach that is suggested in this work.

3.7 Unified version

Summarizing what has been described so far, we can state that

- The global version of the PSO converges fast but can be trapped into local minima when the problem is very hard (results may vary depending on the population initialization or the exploration ability)

- The local version of the PSO can improve the exploration ability of the swarm, but the computational time for the convergence is greater than the one of the global paradigm.

Taking advantage of both the global and local paradigms, a unified PSO version has been introduced in Ref. [75]. In this case, one can define a law according to which the PSO can switch from the local paradigm to the global one. In fact, starting with the local version, the swarm can properly explore the SS and identify the regions that must be exploited. From a comparison of all the regions, the optimizer may find the global optimal position. After that, the global version allows a rapid convergence toward the global minimum.

Referring to Fig. 3.3, the particle velocity is defined as

$$\mathbf{v}_i^{(k+1)} = w^{(k)}\mathbf{v}_i^{(k)} + r_1 c_p\left(\boldsymbol{p}_{best,i}^{(k)} - \mathbf{x}_i^{(k)}\right) + r_2 c_l^{(k)}\left(\boldsymbol{l}_{best,i}^{(k)} - \mathbf{x}_i^{(k)}\right) + r_3 c_g^{(k)}\left(\boldsymbol{g}_{best}^{(k)} - \mathbf{x}_i^{(k)}\right) \tag{3.12}$$

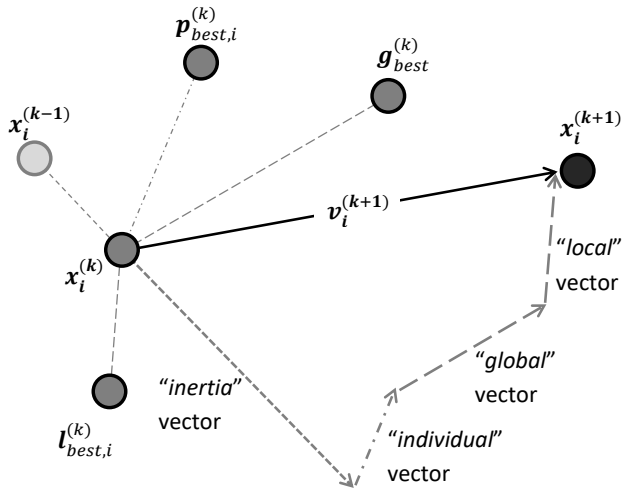


Fig. 3.3: Displacement of a PSO particle in the unified paradigm.

where $c_l^{(k)}$ and $c_l^{(k)}$ change during the optimization process. The switch from the local version to the global version may be implemented in a variety of manners (see Ref. [75]). In this thesis, a linear transition is used. Accordingly, the $c_l^{(k)}$ and $c_g^{(k)}$ weights are given as

$$c_{(\cdot)}^{(k)} = \begin{cases} c_{(\cdot)}^0 - (c_{(\cdot)}^0 - c_{(\cdot)}^f)\dfrac{k-1}{K} & \text{if } \; k \leq K\,, \\ c_{(\cdot)}^f & \text{if } \; k > K\,. \end{cases} \tag{3.13}$$

which is the same linear law used for the inertia weight in Eq. (3.8). In this thesis, whenever the unified version is implemented, we will impose $c_l^0 = 2$, $c_l^f = 0$, $c_g^0 = 0$ and $c_g^f = 2$.

It is noteworthy that for very hard problems where many local minima exist in the SS, the unified version can still make the swarm converge toward local minima. In these cases, the local version is the most suitable paradigm.

3.8 Convergence

Convergence of the swarm towards a stable position with velocity tending to zero is demonstrated in Ref. [68]. The time required for convergence depends on the parameters in Eq. (3.12) and on the number of particles. Nothing guarantees that the point of convergence is the sought optimum.

The decision for stopping the algorithm can depend on several criteria, related to the available problem information, resources, or the ability of the algorithm to attain

further solutions. In [69], several strategies for computing the PSO convergence are reported. Such methods apply not only to PSO, but can be used with most of the metaheuristic algorithms. A brief explanation of these strategies is reported below. In the following, let $\mathbf{x}^*$ be a local or global minimizer of the cost function and

$$J^* = f(\mathbf{x}^*) \tag{3.14}$$

be the corresponding (local or global) minimum. Moreover, $\varepsilon > 0$ is a small user-defined parameter determining the achievement of convergence.

3.8.1 Convergence in search space

If $\|\cdot\|$ denotes a distance measure in the search space, then convergence in SS is defined as:

$$\lim_{k\to\infty} \|\boldsymbol{g}_{best}^{(k)} - \mathbf{x}^*\| = 0. \tag{3.15}$$

Since the available number of iterations is always finite, the convergence criterion can be relaxed as follows: for any desirable accuracy, $\varepsilon > 0$, there is an integer, $k^* > 0$, such that:

$$\|\boldsymbol{g}_{best}^{(k)} - \mathbf{x}^*\| \leq \varepsilon, \forall k \geq k^*. \tag{3.16}$$

Thus, the algorithm is terminated as soon as a solution adequately close to the minimizer is detected.

In practice, the minimizer $\mathbf{x}^*$ is unknown; thus, we can identify convergence by monitoring the gradient at the approximating solutions. However, in order to extract sound conclusions through gradients, the existence of strong mathematical properties (such as continuous differentiability) are required for the objective function. The computation of first-order and second-order derivatives to check the minimizing condition is not always easy for complex problems. In addition, there is no way to distinguish whether the obtained minimizer is the global one, unless additional restrictions (e.g., convexity) are considered in the form of the objective function. PSO and, in general, evolutionary algorithms have been designed to solve problems where the aforementioned required mathematical properties are not necessarily met. Thus, this type of exit condition is of limited practical interest.

3.8.2 Convergence in function values

Convergence can be defined in function values. Indeed, we can reasonably state that PSO has reached convergence if the variation of the cost function is smaller than a user defined threshold. Let us define $\hat{k}$ as the iteration index where the global

best is updated. Accordingly, knowing that $J_g^{(\hat{k}-1)} > J_g^{(\hat{k})} \; \forall \, \hat{k}$, the simple stopping criterion

$$J_g^{(\hat{k}-1)} - J_g^{(\hat{k})} < \varepsilon \tag{3.17}$$

could be defined. However, the stopping criterion in Eq. (3.17) can lead to false stop condition if PSO is not stable around the value $J_g^{(\hat{k})}$. Indeed, it may happen that two successive improvements of the global best particle are very close to each other also in the very first part of the PSO evolution. To increase the reliability of Eq. (3.17), the stopping condition is stated as

$$\delta J = \frac{1}{M} \sum_{i=\hat{k}-M+1}^{\hat{k}} \left(J_g^{(i-1)} - J_g^{(i)} \right) < \varepsilon \tag{3.18}$$

where M is a user defined parameter. Eq. (3.18) evaluates the stopping criterion by an average value of the difference of successive values of J. Finally, Eq. (3.18) can be improved substituting absolute differences of J with relative differences, i.e.

$$\delta J = \frac{1}{M} \sum_{i=\hat{k}-M+1}^{\hat{k}} \frac{J_g^{(i-1)} - J_g^{(i)}}{J_g^{(i)}} < \varepsilon. \tag{3.19}$$

This last formulation makes it easier to adopt the same stopping criterion with completely different optimization problems, as the exit tolerance is related to relative variations instead of absolute variations. Indeed, absolute variations of J can have different orders of magnitude for different optimization problems, whereas relative differences at convergence have more similar values. Such an exit condition allows the PSO to converge until all the particles are within the same neighborhood around the detected minimum.

In the literature there are several optimization problems where the global minimum J^* is a priori known due to the form of the objective function. For example, neural network training is equivalent to the minimization of a function that is usually defined as the summed square-error of the network's output. By construction, this function has the global minimum equal to zero. In such cases, the exit criterion in function values is stated as

$$|J_g^{(\hat{k})} - J^*| \leq \varepsilon. \tag{3.20}$$

3.8.3 Prescribed computational burden

In modern applications, the available time for computation is usually limited. Time critical decisions and on-line control of systems require algorithms that provide satisfactory solutions within very restrictive time frames. An example of such a time-critical problem is represented by autonomous systems implemented on-

board of modern (and even more in future) satellites. For instance, implementing autonomous guidance systems to plan reference maneuvers directly on-board of the satellite requires very efficient algorithms with low computational cost. In fact, on-board computers usually have limited computational performances. Moreover, scheduled queues are usually defined to execute all the different tasks required by the satellite (navigation, control, communication, etc) inside predefined CPU cycles. Therefore, limitations are usually posed on the available computational time (CPU time) for the execution of an algorithm. Interesting discussions about the opportunity to implement real-time optimal control (RTOC) are reported in [27] where such issues are properly considered.

Limitations are also imposed for reasons of comparison. In order to have fair comparisons among algorithms on a specific problem, they must assume the same computational budget. However, a significant issue arises at this point. The time needed for the execution of a program depends heavily on the implementation, programming language, programmer skills, and machine load at the time of execution. Thus, any comparison between two algorithms, in terms of the required CPU time, without taking these factors into consideration is condemned to be biased. For this reason, researchers have made a compromise. The most computationally expensive part in solving a complex problem is expected to be the evaluation of the objective function, which may be computed through complex mathematical procedures or become available directly from experimental devices. Thus, the time required for all function evaluations during the execution of an algorithm is expected to constitute the largest fraction of the overall computation burden. For this purpose, the required number of function evaluations serves very often as a performance measure for optimization algorithms.

Based on the aforementioned discussions, two exit conditions can be defined. Let t_{CPU} be the CPU time (e.g., in seconds) required by the algorithm from the beginning of its execution and

$$N_{eval} = kN_{\mathcal{S}} \tag{3.21}$$

denote the corresponding number of function evaluations required up to step k. It is noteworthy that, in evolutionary algorithms, it is very common to use the number of iterations k instead of function evaluations N_{eval} as a stopping criterion. This is actually one of the best practice with PSO, where population size N_S and number of function evaluations per population member are fixed at each iteration of the algorithm. However, if there is a variable number of function evaluations

per iteration, then only the function evaluations can be used to define the stopping criterion. Accordingly, for PSO the following exit conditions can be defined:

$$t_{CPU} \geq t_{max} \tag{3.22}$$

and,

$$k \geq N_{max}, \tag{3.23}$$

where t_{max} is the maximum available CPU time and N_{max} is the maximum allowed number of PSO iterations. Thus, the algorithm will stop as soon as its CPU time exceeds the available time frame or the number of iterations required so far exceeds an upper limit. For reasons explained above, the condition of Eq. (3.23) is preferred against that of Eq. (3.22). In the following of this thesis, exit condition of Eq. (3.23) is usually employed to end the PSO computation in case that exit condition of Eq. (3.19) is never satisfied for $k \leq N_{max}$.

3.8.4 Search stagnation

Monitoring the progress of an algorithm during the optimization procedure provides insight regarding its efficiency and potential for further improvement of the obtained solutions. The lack of such potential is called search stagnation, and it can be attributed to several factors. In evolutionary algorithms, search stagnation can be identified by monitoring changes of the overall best solution within a specific number of iterations. An algorithm is considered to suffer stagnation if its best solution has not been improved for a number N_{frame} of consecutive iterations, which is defined as a fraction of the maximum number of iterations N_{max}, i.e.

$$N_{frame} = hN_{max}\,,\ h \in (0,1). \tag{3.24}$$

Alternatively, one can identify search stagnation by monitoring diversity of the population, which is usually defined as its spread in the search space. The standard deviation of the population is a commonly used diversity measure. If it falls under a predefined (usually problem-dependent) threshold, then the population is considered to be collapsed on a single point, having limited potential for further improvement. Other features of the PSO algorithm can be used to define diversity. For example, if velocities of all particles become smaller than a threshold, then the swarm can be regarded as immobilized. Thus, its ability for further improvement is limited, and it is questionable whether its further execution can offer any gain.

3.8.5 Selection of the exit condition

There is no general rule for selecting a proper exit condition applicable to all algorithms and problems. Generally, the user has to combine several criteria to ensure that the algorithm is not stalled and worth continuing its execution. From the author's experience, the exit criterion in Eq. (3.19) can be easily implemented for different problems with satisfactory results. Indeed, in the remainder of this thesis the exit condition is implemented as in (3.19) unless stated otherwise. Commonly, exit conditions in function values are combined with condition in Eq. (3.23). In this way, if for some reason no convergence is achieved, PSO stops after a reasonable number of iterations. This is quite useful when dealing with a new optimization problem for the first times, when coding or understanding errors can affect the behavior of the algorithm. In the same way, a measure of search stagnation can help understanding if the population is concentrating around the global best particle or is exploring the search space. For instance, high values of the standard deviation of the population are expected during the exploration phase, whereas small values characterize the exploitation phase.

3.9 Endnotes

In this chapter the Particle Swarm Optimization has been described. Three different paradigms has been presented, global, local and unified. The exploration and exploitation ability of the algorithm have been described and linked to the parameters defining the swarm intelligence algorithm. Coding-simplicity and low computational effort are fundamental characteristics of the algorithm and explain why this approach has been chosen to solve the nonlinear optimization problem in the following chapters. Other metaheuristic algorithms exist in literature, and a comparison with the Differential Evolution will be given in the last chapter of this thesis. In the remaining of this thesis, the Particle Swarm Optimization will be employed to solve the proposed optimal control problems, leading to efficient and reliable solutions.

4

Definition of the Inverse-dynamics Particle Swarm Optimization

> *Order and simplification are the first steps toward the mastery of a subject.*
>
> — **Thomas Mann**
> (German writer)

Abstract

The Inverse-dynamics Particle Swarm Optimization is the main topic of this thesis. This optimization method has been developed to find near-optimal solutions to optimal control problems in a straightforward way with low computational effort. The assumptions upon which the optimizer is based are described, as well as the main important characteristics required for the implementation of the algorithm.

Nomenclature

$\boldsymbol{x}(t)$	=	State of the dynamical system	$\boldsymbol{u}(t)$	=	Control of the dynamical system
$\boldsymbol{y}(t)$	=	Flat output	N_u	=	Control dimension
J	=	Cost functional	E	=	End-point cost functional
F	=	Running cost functional	t	=	Time (s)
$\boldsymbol{\mathcal{K}}$	=	Knot vector	κ	=	Component of the knot vector
$\mathbf{x}^{(k)}$	=	PSO particle	$\tilde{(\cdot)}$	=	Parameter defining the B-spline
$\mathcal{B}$	=	B-spline	λ	=	B-spline independent variable
$N_{\mathcal{P}}$	=	Number of B-spline polynomials	$\mathcal{N}$	=	B-spline basis function
$(\cdot)^N$	=	B-spline approximation	$\mathcal{D}$	=	B-spline degree
$\mathbf{U}$	=	B-spline control point	$\dot{(\cdot)}$	=	First time derivative
$\ddot{(\cdot)}$	=	Second time derivative	$\overline{J}$	=	Extended cost functional
$(\cdot)'$	=	λ-derivative	γ_f	=	Graph of the generic function f
N_T	=	Discretization points	k	=	PSO iteration
$\mathcal{P}$	=	PC penalty functions	$(\cdot)_{0/f}$	=	Initial/final time value
N_{viol}	=	Number of violated constraints	m	=	Knot vector length parameter
π, μ	=	Penalty function weights	Δ	=	Constraint tolerance
$\bar{k}$	=	Exterior cycle index	δ	=	Decreasing tolerance parameter

N_δ	=	Decreasing tolerance parameter	$\delta\overline{J}_g$	=	Convergence index
$\mathscr{Y}_B$	=	Set of BC-compliant functions	$\mathcal{Y}_B$	=	Set of BC-compliant parameters
$\mathbb{Y}$	=	Flat output space	ε_c	=	Convergence tolerance

4.1 Introduction

The Inverse Dynamics Particle Swarm Optimization (IPSO) is a numerical technique to solve optimal control problems first presented by Spiller *et al.* in 2015 (see Ref. [1]) and improved in successive works (see Refs. [2, 3, 8, 9]). The main characteristics of IPSO are reported in [4] and are summarized below:

1. A differential flatness parametrization of the optimal control problem is employed, with the straightforward advantage of reducing to the minimum number the independent optimization parameters.

2. The approximation of the flat parameter is accomplished with B-spline curves [76, 77]. A variable time-mesh is introduced describing the time function as a B-spline. Hence, each component of the flat output is described with a 2 degree-of-freedom curve. This feature is referred to as improved B-spline approximation.

3. The optimal parameters of the transcribed BOCP are searched for with the Particle Swarm Optimization [66, 67].

4. The differential flatness parametrization allows one to satisfy a-priori the boundary conditions, eliminating the related equality constraints.

5. Adaptive decreasing tolerances (see Ref. [3, 8]) are employed to take into account inequality constraints. This technique helps the optimizer exploring the search space and comparing several local minima. The IPSO implements the refined law described in Sec. 4.5.2.

The IPSO is a technique that can be employed for all the optimization problems that accept a differential flatness formulation. To the best of the author knowledge, most of the problems of engineering interest may be cast in a differential flatness formulation.

In the following of this chapter, we will introduce several approaches for implementing the B-splines approximation and the decreasing tolerances. To follow the historical development of the IPSO, two versions are identified:

- **Basic IPSO**: this version is based on the *basic* B-spline approximation and the *basic* adaptive decreasing tolerance law.

- **Improved IPSO**: this version is based on the *improved* B-spline approximation and the *improved* adaptive decreasing tolerance law.

Details about the B-spline approximation approaches and the adaptive decreasing tolerance laws are given in Sec. 4.2 and Sec. 4.5.2, respectively. The reader should note that the improved IPSO represents the current version of IPSO used in the most recent author's papers. Accordingly, in this work, the acronym IPSO will always refer to the *improved* implementation when it is not explicitly put the adjective "basic".

The acronym IPSO comes from the combination of the inverse dynamics approach (in the form of a differential flatness parametrization) and the Particle Swarm Optimization. As it will explained in detail in Chapter 6, the combination of this two features allows one to solve complicated nonconvex optimal control problems.

The general problem that is addressed by the IPSO has been described in Sec. 2.6 and is summarized here for clarity. Given the flat output $\boldsymbol{y}(\,\cdot\,) \in \mathbb{R}^{N_u}$ and having

$$\boldsymbol{x} = \boldsymbol{a}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, ..., \boldsymbol{y}^{(\beta)}\right) \;, \quad \boldsymbol{u} = \boldsymbol{b}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, ..., \boldsymbol{y}^{(\beta+1)}\right), \quad \beta \in \mathbb{N} \tag{4.1}$$

solve the following problem,

$$\begin{gathered}
\text{Find } \boldsymbol{y}(t) : t \to \mathbb{Y} \subset \mathbb{R}^{N_u},\, t_f \in \mathbb{R} \\
\text{minimizing} \\
J[\boldsymbol{y}(\cdot), t_f] = E(\boldsymbol{y}(t_0), \{\boldsymbol{y}^{(\beta)}(t_0)\}, \boldsymbol{y}(t_f), \{\boldsymbol{y}^{(\beta)}(t_f)\}, t_f) \\
+ \int_{t_0}^{t_f} F\left(\boldsymbol{y}(t), \{\boldsymbol{y}^{(\beta)}(t)\}, t\right) dt \\
\text{subject to, } \forall t \in [t_0, t_f] \\
\text{Boundary constraints:} \quad \mathbf{b}(\boldsymbol{y}(t_0), \{\boldsymbol{y}^{(\beta)}(t_0)\}, \boldsymbol{y}(t_f), \{\boldsymbol{y}^{(\beta)}(t_f)\}, t_f) = \mathbf{0}, \\
\text{Path Constraints:} \quad \mathbf{p}(\boldsymbol{y}(t), \{\boldsymbol{y}^{(\beta+1)}(t)\}, t) \leq \mathbf{0}.
\end{gathered} \tag{4.2}$$

In the following chapters of this thesis, we will refer to the Feasible Search Space (FSS) as the set of solutions which satisfy all the end-point and path constraints.

The remainder of this chapter is organized as follows. In Sec. 4.2 an outline of the most important features of the B-spline curves is given, explaining the difference between the basic and the improved approximation techniques. In Sec. 4.4 the

global search ability of the IPSO is related to using the unified PSO paradigm. In Sec. 4.3 the technique for the a-priori satisfaction of the boundary constraints is explained, whereas in Sec. 4.5 the constraint handling technique is described. Finally, conclusion notes are given in Sec. 4.7.

4.2 Curves approximation with B-splines

B-spline curves have been introduced in Ref. [76, 77, 78] and are a well-known instrument in computer aided engineering design.

The reason why B-splines curves have been chosen to approximate the flat output is that they are spline functions that have minimal support with respect to a given degree, smoothness, and domain partition (see Ref. [79]). Consequently, B-spline curves can guarantee good approximation of the optimal flat output history with a reduced number of optimization parameters. A comparison of B-spline curves with Chabyshev polynomials is carried out in Chapter 9, where it is numerically demonstrated that the the former approach guarantees better results then the latter. The B-spline approximation has been also proposed in Ref. [45] associated to a collocation method.

In the following of this chapter functions will be defined upon input arguments and parameters. The following syntax will be used:

$$function\_name(1^{st}\ variable, 2^{nd}\ variable, ...\ ;\ 1^{st}\ parameter, 2^{nd}\ parameter, ...)$$

A semicolon is used to separate variables from parameters. Quite often, the terms variables and parameters are used interchangeably, but with a semicolon the meaning is that we are defining a function of the parameters that returns a function of the variables. For example, the syntax $f(x_1, x_2, \dots; p_1, p_2, \dots)$ is used to mean that, by supplying the parameters $(p_1, p_2, \dots)$, a new function is created whose arguments are $(x_1, x_2, \dots)$.

For the flat output $\boldsymbol{y}$, the B-spline approximation $\mathcal{B}(\lambda; \tilde{\boldsymbol{y}}, \boldsymbol{\mathcal{K}})$ is defined upon a strictly increasing independent variable $0 \leq \lambda \leq 1$ and depends on the coefficient vector $\tilde{\boldsymbol{y}}$ and the *knot vector* $\boldsymbol{\mathcal{K}}$ with $m+1$ components defined as

$$\boldsymbol{\mathcal{K}} = \{\kappa_i,\ i = 0, ..., m \,|\, \kappa_0 = 0, \kappa_m = 1, \kappa_i \leq \kappa_{i+1}\}\,. \tag{4.3}$$

A B-spline combining $N_{\mathcal{P}}$ polynomials of $\mathcal{D}^{th}$ degree satisfies $m = N_{\mathcal{P}} + \mathcal{D}$. The generic component y of the flat output (the subscript for the jth component of the flat output is not reported to avoid confusion) vector can be approximated as

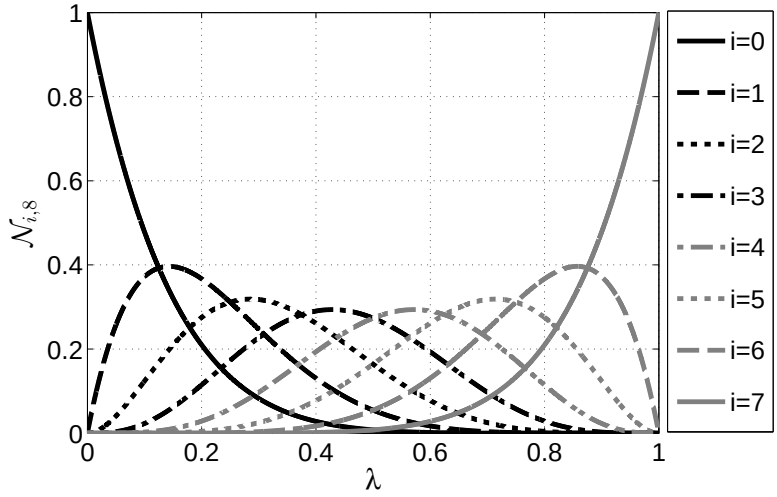


Fig. 4.1: B-spline basis functions $\mathcal{N}_{i,\mathcal{D}}$ with $N_\mathcal{P} = \mathcal{D} + 1 = 8$.

a B-spline $\mathcal{B}(\lambda; \tilde{\boldsymbol{y}}, \boldsymbol{\mathcal{K}})$ function. Indeed, the approximated flat parameter $y^N(\lambda)$ is defined as

$$y^N(\lambda) : \lambda \mapsto \mathcal{B}(\lambda; \tilde{\boldsymbol{y}}, \boldsymbol{\mathcal{K}}). \tag{4.4}$$

Hence, given the optimization parameters in $\tilde{\boldsymbol{y}}$ and the fixed vector $\boldsymbol{\mathcal{K}}$, the flat parameter is a function of the independent variable λ. In further details, $y^N(\lambda)$ is evaluated as

$$y^N(\lambda) = \mathcal{B}(\lambda; \tilde{\boldsymbol{y}}, \boldsymbol{\mathcal{K}}) = \sum_{i=0}^{N_\mathcal{P}-1} \tilde{y}_i \mathcal{N}_{i,\mathcal{D}}(\lambda; \boldsymbol{\mathcal{K}}), \tag{4.5}$$

where the basis functions $\mathcal{N}_{i,d}(\lambda; \boldsymbol{\mathcal{K}})$ are defined through the *Cox-de Boor recursion formula*. For $d = 0$,

$$\mathcal{N}_{i,0}(\lambda; \boldsymbol{\mathcal{K}}) = \begin{cases} 1 & \text{if } \kappa_i \leq \lambda < \kappa_{i+1}\,, \\ 0 & \text{otherwise}\,, \end{cases} \tag{4.6}$$

where $i = 0, ..., m-1$. For $1 \leq d \leq \mathcal{D}$,

$$\mathcal{N}_{i,d}(\lambda; \boldsymbol{\mathcal{K}}) = \frac{\lambda - \kappa_i}{\kappa_{i+d-1} - \kappa_i} \mathcal{N}_{i,d-1}(\lambda; \boldsymbol{\mathcal{K}}) + \frac{\kappa_{i+d} - \lambda}{\kappa_{i+d} - \kappa_{i+1}} \mathcal{N}_{i+1,d-1}(\lambda; \boldsymbol{\mathcal{K}})\,, \tag{4.7}$$

where $i = 0, ..., m-d$.

An example of the basis functions obtained with $N_\mathcal{P} = \mathcal{D} + 1 = 8$ is reported in Fig. 4.1. Note that the range of the basis functions is $[0, 1]$.

Full advantage of the B-spline is taken approximating the trajectory as a *curve*[8, 2, 3, 9], not as a *function*. To build a B-spline curve, a polyline defined over the control points $\mathbf{U}_j = [\tilde{t}_j t_f, \tilde{y}_j]$, $j = 0, ..., N_\mathcal{P} - 1$, is introduced. In this case, the time is not an independent variable (usually identified with λ, once normalized) but is a

real-valued function evaluated as a B-spline. Introducing the time coefficient vector $\tilde{\boldsymbol{t}} = [\tilde{t}_{i,1} = 0, \tilde{t}_{i,2} > \tilde{t}_{i,1}, ..., \tilde{t}_{i,N_{\mathcal{P}}-1} = 1]$, the time is defined as

$$t^N(\lambda) : \lambda \mapsto t_f \mathcal{B}(\lambda; \tilde{\boldsymbol{t}}, \boldsymbol{\mathcal{K}}) \tag{4.8}$$

and it is evaluated as

$$t^N(\lambda) = t_f \mathcal{B}(\lambda; \tilde{\boldsymbol{t}}, \boldsymbol{\mathcal{K}}) = t_f \sum_{i=0}^{N_{\mathcal{P}}-1} \tilde{t}_i \, \mathcal{N}_{i,\mathcal{D}}(\lambda; \boldsymbol{\mathcal{K}}) \, . \tag{4.9}$$

As a consequence, the independent variable used for the description of the problem is λ. Hence, $\forall \, t^N(\lambda) \in [0, t_f]$, the flat parameter time-history is a function described by the two-dimensional graph

$$\boldsymbol{\gamma}_y(\lambda) = \left\{ \left(t^N(\lambda), y^N(\lambda) \right), \lambda \in [0,1] \right\} . \tag{4.10}$$

An explicit function $\bar{y}^N\left(t^N\right)$ is not easily evaluable, so the B-spline approximation of the flat parameter expressed as a function of the time is given in a tabular form (as a series of pairs $(t^N(\lambda), y^N(\lambda))$). The first time derivative of y is evaluated as

$$\dot{y}^N(\lambda) = \frac{d\mathcal{B}(\lambda; \tilde{\boldsymbol{y}}, \boldsymbol{\mathcal{K}})}{dt^N} = \frac{\partial \mathcal{B}(\lambda; \tilde{\boldsymbol{y}}, \boldsymbol{\mathcal{K}})}{\partial \lambda} \frac{\partial \lambda}{\partial t^N} \tag{4.11}$$

and, by means of Eq. (4.9) and defining $\mathcal{B}'_y = \partial \mathcal{B}(\lambda; \tilde{\boldsymbol{y}}, \boldsymbol{\mathcal{K}})/\partial \lambda$, $\mathcal{B}'_t = \partial \mathcal{B}(\lambda; \tilde{\boldsymbol{t}}, \boldsymbol{\mathcal{K}})/\partial \lambda$,

$$\dot{y}^N(\lambda) = \frac{\partial \mathcal{B}(\lambda; \tilde{\boldsymbol{y}}, \boldsymbol{\mathcal{K}})}{\partial \lambda} \frac{\partial \lambda}{\partial \mathcal{B}(\lambda; \tilde{\boldsymbol{t}}, \boldsymbol{\mathcal{K}})} = \frac{\mathcal{B}'_y}{\mathcal{B}'_t} \, . \tag{4.12}$$

Analogously, the second time derivative is given by

$$\ddot{y}^N(\lambda) = \frac{d^2 \mathcal{B}(\lambda; \tilde{\boldsymbol{y}}, \boldsymbol{\mathcal{K}})}{d(t^N)^2} = \frac{\mathcal{B}''_y \mathcal{B}'_t - \mathcal{B}''_t \mathcal{B}'_y}{\mathcal{B}'^3_t} \, . \tag{4.13}$$

The terms $\mathcal{B}'_y$ and $\mathcal{B}''_y$ (similarly $\mathcal{B}'_t$ and $\mathcal{B}''_t$) are given by

$$\mathcal{B}'_y = \sum_{i=0}^{N_{\mathcal{P}}-2} \tilde{y}_i^{(1)} \mathcal{N}_{i+1,\mathcal{D}-1}(\lambda; \boldsymbol{\mathcal{K}}) \, , \quad \mathcal{B}''_y = \sum_{i=0}^{N_{\mathcal{P}}-3} \tilde{y}_i^{(2)} \mathcal{N}_{i+2,\mathcal{D}-2}(\lambda; \boldsymbol{\mathcal{K}}) \, , \tag{4.14}$$

where the coefficients $\tilde{y}_i^{(1)}$ and $\tilde{y}_i^{(2)}$ are the following finite differences:

$$\tilde{y}_i^{(1)} = \mathcal{D} \frac{\tilde{y}_{i+1} - \tilde{y}_i}{\kappa_{i+\mathcal{D}} - \kappa_i} \, , \quad \tilde{y}_i^{(2)} = (\mathcal{D} - 1) \frac{\tilde{y}'_{i+1} - \tilde{y}'_i}{\kappa_{i+\mathcal{D}-1} - \kappa_i} \, . \tag{4.15}$$

The B-spline approximation of a generic curve is shown in Fig. 4.2. It is noteworthy that the control points may move 1) in the vertical direction, which represent a variation of the approximated parameter (either state or control or flat output), and 2) in the horizontal direction, which represent a variation of the time mesh.

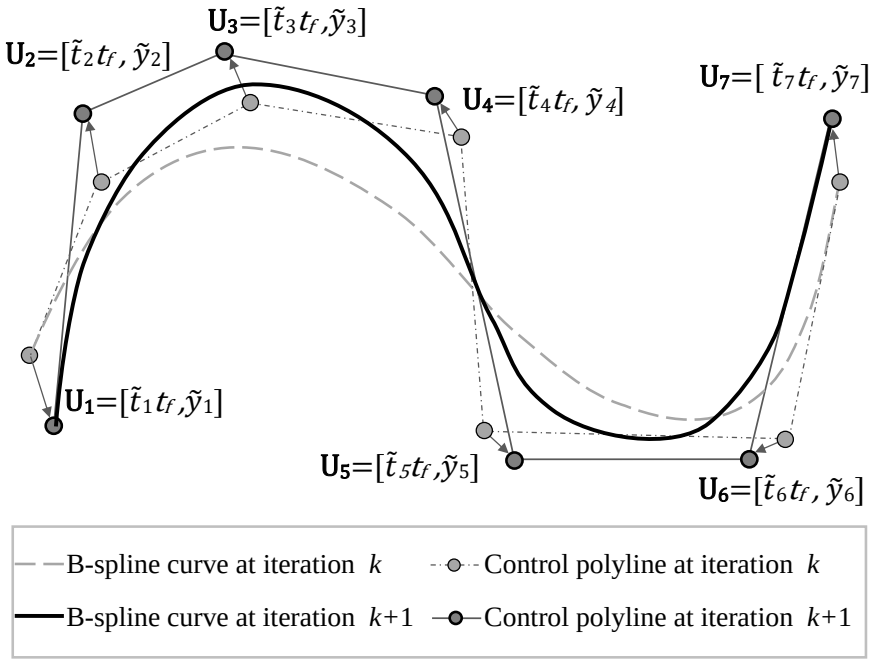


Fig. 4.2: Improved B-splines curve approximation.

Note that, when Eq. (4.9) is modified using $t = t_f\lambda$, Eqs. (4.12)-(4.13) converge to

$$\dot{y}^N(\lambda) = t_f^{-1}\mathcal{B}' \,, \qquad \ddot{y}^N(\lambda) = t_f^{-2}\mathcal{B}''. \tag{4.16}$$

In this case, we can recognize t as the independent variable such that $y(t) = y^N(t/t_f)$ and the flat parameter time-history is a function described by the two-dimensional graph

$$\gamma_y(\lambda) = \left\{ \left(t_f\lambda, y^N(\lambda) \right), \lambda \in [0,1] \right\}. \tag{4.17}$$

The method described from Eq. (4.5) to Eq. (4.13) can be referred to as *improved* B-spline approximation, opposed to the *basic* B-spline approximation given by Eq. (4.16). The advantages of the improved model is that the curve is shaped with 2 degrees of freedom, whereas the basic model is described by 1 degree of freedom. With regard to Fig. 4.2, the control points of the basic B-spline approximation can only move in the vertical direction.

In the following of this thesis, second order three-dimensional differential equations will be employed to describe the dynamical systems involved in the proposed OCPs. From Eq. (4.1) and Eqs. (4.12)-(4.15), the generic component u_i, $i = 1, ..., N_u$, of external control is obtained as a closed-form solution in terms of the B-spline approximated flat parameters $y_i^N(\lambda)$, $i = 1, ..., N_u$. Using the improved B-spline approximation, however, an explicit time-function of the control is not

identified. On the contrary, the control can be easily described in terms of the curves (t, u_i) parametrized by λ, i.e.

$$\begin{aligned}\boldsymbol{\gamma}_u(\lambda) &= \left\{\left(t^N(\lambda), u_i(\lambda)\right), \lambda \in [0,1]\right\} \\ &= \left\{\left(t^N(\lambda), b_i\left(y_1^N(\lambda), ..., y_{N_u}^N(\lambda)\right)\right), \lambda \in [0,1]\right\}.\end{aligned} \tag{4.18}$$

4.3 A-priori satisfaction of boundary constraints

Using an inverse dynamics approach with a differential flatness parametrization, the end-point condition concerning the state in t_0 and t_f may be imposed a-priori using a proper polynomial approximation. The B-spline approximation allows the user to impose the values of the initial and final flat output. Accordingly, in the sequel the *clamped* B-spline will be employed: in this case, the curve passes through $\mathbf{A_0}$ and $\mathbf{A}_{\boldsymbol{N_\mathcal{P}-1}}$ and it is obtained using *non-uniform knot points* given by

$$\begin{aligned}&\kappa_j = 0 && if\ \ 0 \leq j \leq \mathcal{D}, \\ &\kappa_j = \frac{j-\mathcal{D}}{N_\mathcal{P}-\mathcal{D}} && if\ \ \mathcal{D} < j \leq N_\mathcal{P}-1, \\ &\kappa_j = 1 && if\ \ N_\mathcal{P}-1 < j \leq m.\end{aligned} \tag{4.19}$$

For a clamped B-splines it is verified that $\mathcal{N}_{i,d}(0;\boldsymbol{\mathcal{K}}) = \delta_{i,\mathcal{D}-d}$ and $\mathcal{N}_{i,d}(1;\boldsymbol{\mathcal{K}}) = \delta_{i,\mathcal{D}+d}$, where $\delta_{i,j}$ is the Kronecker delta. Hence, from Eq. (4.15), it can be proved that the clamped B-spline is tangent to the first and to the last leg of the control polyline. Accordingly, initial and final conditions for $y(t)$ are imposed setting

$$\begin{aligned}&\tilde{y}_0 = y(t_0), && \tilde{y}_1 = \tilde{y}_0 + (\tilde{t}_1 - \tilde{t}_0)\dot{y}(t_0), \\ &\tilde{y}_{N_\mathcal{P}-1} = y(t_f), && \tilde{y}_{N_\mathcal{P}-2} = \tilde{y}_{N_\mathcal{P}-1} - (\tilde{t}_{N_\mathcal{P}-1} - \tilde{t}_{N_\mathcal{P}-2})\dot{y}(t_f).\end{aligned} \tag{4.20}$$

The reader should remember that $\tilde{y}_j$, $j = 1, ..., N_\mathcal{P}$, are the optimization variables of the problem. With the conditions in Eq. (4.20) we reduce the number of unknown parameters and we guarantee from the beginning of the optimization process the satisfaction of the boundary conditions. This is an important advantage with respect to direct dynamics methods. In fact, in that case only initial conditions may be imposed at the beginning of the integration process, whereas final condition can be satisfied only accepting a final error within a user-defined tolerance. By imposing boundary conditions before the numerical optimization, the transcribed problem does no more require to consider boundary constraints.

To underline that a flat output approximation $\boldsymbol{y}^N$ satisfies the boundary constraints by means of Eq. (4.20), the set of BC-compliant functions

$$\mathscr{Y}_B = \{y^N(\lambda) = \mathcal{B}(\lambda; \tilde{\boldsymbol{y}}, \boldsymbol{\mathcal{K}}) : \tilde{\boldsymbol{y}} \text{ is defined as in Eq. (4.20)}\} \tag{4.21}$$

is introduced. Accordingly, the statement $\boldsymbol{y}^N \in \mathbb{Y}_B$ means that $\boldsymbol{y}^N$ satisfies the boundary constraints. In the same way, we can define the set of BC-compliant parameters

$$\mathcal{Y}_B = \{\tilde{\boldsymbol{y}} : \tilde{\boldsymbol{y}} \text{ is defined as in Eq. (4.20)}\} \tag{4.22}$$

such that $\tilde{\boldsymbol{y}} \in \mathcal{Y}_B$ means that the B-spline parameters are defined in order to guarantee $\boldsymbol{y}^N \in \mathbb{Y}_B$.

4.4 Particle Swarm transcription

Having introduced the clamped B-spline and the a-priori satisfaction of the end-point constraints, the optimization problem addressed by the IPSO can be simplified to:

$$\begin{gathered}
\text{Find } \boldsymbol{y}(t) : t \to \mathbb{Y} = \mathbb{R}^{N_u} \text{ such that } \boldsymbol{y} \in \mathscr{Y}_B,\ t_f \in \mathbb{R} \\
\text{minimizing} \\
J[\boldsymbol{y}(\cdot), t_f] = E(\boldsymbol{y}(t_0), \{\boldsymbol{y}^{(\beta)}(t_0)\}, \boldsymbol{y}(t_f), \{\boldsymbol{y}^{(\beta)}(t_f)\}, t_f) \\
+ \int_{t_0}^{t_f} F\left(\boldsymbol{y}(t), \{\boldsymbol{y}^{(\beta)}(t)\}, t\right) dt \\
\text{subject to, } \forall t \in [t_0, t_f] \\
\text{Path Constraints:} \quad \mathbf{p}(\boldsymbol{y}(t), \{\boldsymbol{y}^{(\beta+1)}(t)\}, t) \leq \mathbf{0}.
\end{gathered} \tag{4.23}$$

The basic paradigms of the PSO have been described in Chapter 3. The IPSO employees the PSO to exploit its global search ability for hard optimization problems affected by nonconvexity issues.

The IPSO is based on the application of the unified paradigm. This implementation allows the optimizer to take advantage of the enhanced exploration ability of the local PSO version. Moreover, at the end of the optimization process, the exploitation characteristic of the global PSO version are used to refine the final solution.

It is to be noted that for extremely difficult problems, affected by several distinct local minima, the local paradigm may perform better than the unified paradigm. In the following, it will explicitly reported what PSO is employed for each problem.

To take advantage of the exploration and exploitation abilities of PSO, the unified paradigm described in Sec. 3.7 is employed. Accordingly, the coefficients c_l and c_g in Eq. (3.12) vary during the optimization process in order to allow the swarm to explore the search space in the beginning of the optimization process and then converge toward the optimal solution at the end of the evolution. In particular, the global and local coefficients follow the linear law expressed in Eq. (3.13).

For the implementation of the IPSO, the particle can be defined in different ways depending on what type of B-pline approximation is used. We remind the reader that the flat output $\boldsymbol{y}$ is defined in $\mathbb{R}^{N_u}$. When the *improved* B-spline approximation is used, the number of time meshes is equal to N_u, as every component of the flat output has its own time mesh.

IPSO particle with *basic* B-spline approximation In this case, the basic B-spline approximation is employed and a unique linear time mesh is introduced. Therefore, the minimum number of parameters is employed. For a fixed-time optimal control problem, the IPSO particle is defined as

$$\mathbf{x} = [\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_{N_u}] \in \mathbb{R}^{N_u N_{\mathcal{P}}}, \tag{4.24}$$

whereas for a free-time problem the particle is

$$\mathbf{x} = [\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_{N_u}, t_f] \in \mathbb{R}^{N_u N_{\mathcal{P}}+1}. \tag{4.25}$$

IPSO particle with *improved* B-spline approximation In this case, the improved B-spline approximation is employed and the related time mesh is introduced, increasing the dimension of the particle with respect to the previous case. For a fixed-time optimal control problem, the IPSO particle is defined as

$$\mathbf{x} = [\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_{N_u}, \tilde{\boldsymbol{t}}_1, ..., \tilde{\boldsymbol{t}}_{N_u}] \in \mathbb{R}^{2N_u N_{\mathcal{P}}}, \tag{4.26}$$

whereas for a free-time problem the particle is

$$\mathbf{x} = [\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_{N_u}, \tilde{\boldsymbol{t}}_1, ..., \tilde{\boldsymbol{t}}_{N_u}, t_f] \in \mathbb{R}^{2N_u N_{\mathcal{P}}+1}. \tag{4.27}$$

In the general case, defining the vectors $\widetilde{\boldsymbol{Y}} = [\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_{N_u}]$ and $\widetilde{\boldsymbol{T}} = [\tilde{\boldsymbol{t}}_1, ..., \tilde{\boldsymbol{t}}_{N_u}]$, the PSO particle is

$$\mathbf{x} = [\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, t_f] \in \mathbb{R}^{2N_u N_{\mathcal{P}}+1}. \tag{4.28}$$

In Sec. 4.2 we have seen that the flat parameter history and its time derivatives are evaluated as a function of the optimization parameters $\tilde{\boldsymbol{y}}$. Similarly, the time is

parameterized by $\tilde{\boldsymbol{t}}$. As a consequence, the transcribed optimization problem can be written in terms of the quantities inside the PSO particle $\mathbf{x}$ as

$$
\begin{aligned}
&\text{Find } \mathbf{x} = [\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, t_f] \in \mathbb{R}^{2N_u N_{\mathcal{P}}+1} \text{ such that } \tilde{\boldsymbol{y}}_i \in \mathcal{Y}_B,\ \forall i = 1, ..., N_u \\
&\qquad\qquad \text{minimizing} \\
&\qquad J^N(\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, t_f) = E^N(\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, t_f) + I^N(\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, \lambda; t_0, t_f) \qquad (4.29) \\
&\qquad\qquad \text{subject to, } \forall t \in [t_0, t_f] \\
&\text{Path Constraints:} \quad \boldsymbol{p}^N(\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, \lambda) \leq \mathbf{0} \qquad \forall \lambda \in [0, 1].
\end{aligned}
$$

In Eq. (4.29), the terms E^N and I^N are the numerical transcribed counterparts of the end-point cost function E and the integral of the running cost function F. The function $\boldsymbol{p}^N$ is the path constraint evaluated as a function of the optimization parameters.

4.5 Constraint handling technique

Dealing with constrained optimization, the performance index must be properly modified to take into account the alteration of the SS induced by the constraints. Hence, an extended performance index $\overline{J}$ is defined as the summation of the original cost functional J (i.e., the goal of the problem, which may be minimum-time, minimum-energy or minimum-effort) and the penalty functions related to the constraints. The exterior penalty function described in Sec. 1.3.2 is employed. This means that, considering only path inequality constraints, for every constraint $p(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq 0$, $c : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R} \mapsto \mathbb{R}$, a scalar penalty function P is introduced that penalizes the solution if and only if the constraint p is violated.

To evaluate the numerical solution of the optimal control problem, the performance index is evaluated after the transcription of the original problem. Accordingly, the approximated extended cost function $\overline{J}^N$ is evaluated. The difference between $\overline{J}$ and $\overline{J}^N$ is that the former is defined over the state and control functions whereas the latter is defined by means of the approximated state and control functions and depends explicitly on the optimization parameters.

It is noteworthy that, for the IPSO, only inequality constraints must be taken into account in the cost function. In fact, as explained in Sec. 4.3, the equality constraints related to the end-point conditions are automatically satisfied and do not need to be imposed inside the optimization problem. The inequality path constraints may

concern both the state and the control (in the latter case, they are also known as control constraint).

For numerical reasons, the values of the independent variable λ (introduced in Sec. 4.2) are discretized such that $\lambda_j = j/N_T$, $j = 0, ..., N_T$. The cost function J^N changes depending on the goal of the maneuver, i.e.:

- For minimum-time maneuvers $I^N = 0$ and $E^N \neq 0$, i.e.

$$J^N = t_f. \tag{4.30}$$

  Note that in this case there not formal difference between J and J^N.

- For minimum-effort maneuvers $E^N = 0$ and $I^N \neq 0$, i.e.

$$J^N = \frac{t_f - t_0}{2N_T} \sum_{i=1}^{3} \sum_{j=1}^{N_T} \left(|u_i|(\lambda_j - 1) + |u_i|(\lambda_j) \right). \tag{4.31}$$

- For minimum-energy maneuvers $E^N = 0$ and $I^N \neq 0$, i.e.

$$J^N = \frac{t_f - t_0}{2N_T} \sum_{i=1}^{3} \sum_{j=1}^{N_T} \left(u_i^2(\lambda_j - 1) + u_i^2(\lambda_j) \right). \tag{4.32}$$

In Eq. (4.31) and Eq. (4.32), the trapezoidal numerical integration has been employed to approximate Eq. (6.37) and Eq. (6.38), respectively.

The approximated extended cost function $\overline{J}^N$ is evaluated adding to J^N the exterior penalty functions related to the path constraints. Supposing N_C constraints, the PSO performance index is thus given by (see Ref. [1, 8])

$$\overline{J}^N = J^N + \sum_{i=1}^{N_C} \sum_{j=0}^{N_T} \pi_i \mathcal{P}_i(\lambda_j) + \mu \, N_{viol} \,, \tag{4.33}$$

where π_i and μ are user-defined constant weights and N_{viol} is the number of violated constraints. $\mathcal{P}_i$ is the penalty function associated with the constraint p_i.

The IPSO method handles the path constraints using a *relaxation* technique. In fact, the generic inequality constraint $p_i^N(\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, \lambda_j) - \alpha_i \leq 0$ is *relaxed* and treated as $|p_i^N(\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, \lambda_j) - \alpha_i| \leq \Delta_i$. The penalty function P_i may be evaluated in two different ways:

1. $\mathcal{P}_i \in \mathbb{N}$,

$$\mathcal{P}_i(\lambda_j) = \begin{cases} 0 & \text{if } |p_i^N\left(\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, \lambda_j\right)) - \alpha_i| \leq \Delta_i\,, \\ 1 & \text{otherwise.} \end{cases} \tag{4.34}$$

2. $\mathcal{P}_i \in \mathbb{R}$,

$$\mathcal{P}_i(\lambda_j) = \begin{cases} 0 & \text{if } |p_i\left(\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, \lambda_j\right) - \alpha_i| \leq \Delta_i\,, \\ p_i(\widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{T}}, \lambda_j) - \alpha_i - \Delta_i & \text{otherwise.} \end{cases} \tag{4.35}$$

The difference between the two formulations is that the generic weight π_i attains different values in order to give consistent values of the penalty functions.

When the penalty functions will be defined in the numerical applications reported in the following chapters, the path constraints penalty functions may be splitted into those related to the state (or the flat output) and those related to the control. However, the strategy employed to define the penalty function will always be the one defined in this section.

4.5.1 Exterior and interior cycles

A solution to the proposed problem would be considered optimal if and only if the aforementioned tolerances Δ_i were zero. However, two considerations are in order:

- Δ_i related to equality constraints cannot be set to zero due to the numerical integration errors. It means that the optimality of the solution will be associated with *relaxed* final conditions with Δ_i close but not equal to zero.

- The PSO method suffers from convergence problems if tolerances are set equal or very close to zero from the beginning of the algorithm.

The first issue has been already taken into account. In fact, equality constraints are automatically satisfied thanks to the properties of the clamped B-spline. Note that, when all the inequality constraints are satisfied and the maneuver is completely feasible, the extended cost function is equal to J_0. Since the IPSO is able to satisfy all the constraints in few iterations (as it will be shown in the chapters concerning the numerical results), this means that the introduction of the extended cost function does not perturb the original problem, but it does restrict the search to only feasible maneuvers in the very first part of the search operation. On the contrary, when a direct dynamics approach is employed, the equality constraints associated to the

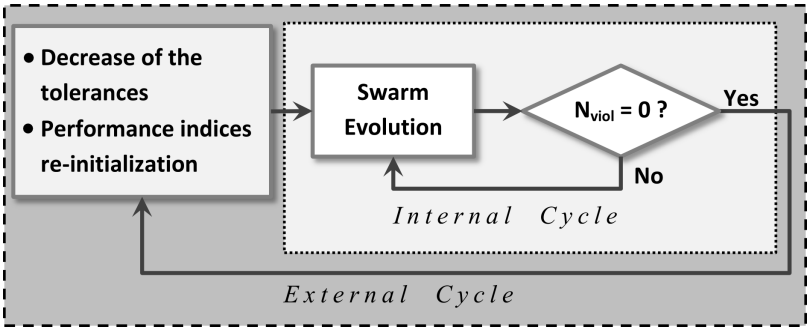


Fig. 4.3: IPSO External and internal cycle.

end-point conditions are never completely satisfied, and the optimal numerical value of $\overline{J}$ will only be an approximation of the optimal value of J.

With regard to the second issue, the penalty functions $\mathcal{P}_i$ are designed with *adaptive decreasing tolerances* Δ_i (see Ref. [8]) to help the swarm convergence. The tolerances change their values if and only if the global best particle in the swarm satisfies all the imposed constraints, i.e. if and only if $N_{viol}^{(g_{best})}=0$. Accordingly, the tolerances decrease in an *external* loop if and only if the evolution of the swarm in the *internal* loop has led to $N_{viol}^{(g_{best})}=0$. In the *external* while loop all the performance indices are reinitialized since the swarm is going to explore a new search environment where the previous solution is no longer valid. This technique is shown in a block diagram in Fig. 4.3.

4.5.2 Adaptive decreasing tolerances

Basic law First, the decreasing tolerance technique employed in Ref. [1] is reported, where a piecewise linear decreasing law was used. The intervals of the piecewise linear function are defined as a function of the tolerance itself. Let us refer to the steps where $N_{viol}=0$ as $\bar{k}$. The generic tolerance $\Delta_{(\cdot)}$ is therefor updated as

$$\Delta_{(\cdot)}^{(\bar{k}+1)} = \Delta_{(\cdot)}^{(\bar{k})} - 10^{-\xi} \tag{4.36}$$

where ξ is selected to satisfy $10^{-\xi} < \Delta_{(\cdot)}^{(\bar{k})} \leq 10^{-\xi+1}$ with $\xi \in \mathbb{N}$. For inequality constraint, after a certain user-defined value $\bar{k}^*$, the tolerance may be set to zero.

Refined law Instead of the piecewise decreasing law, in successive works a more effective law based on only two parameters has been introduced. Starting from user-defined and problem-dependent initial values, the tolerances decrease according to the following law:

$$\Delta_{(\cdot)}^{(\bar{k}+1)} = \Delta_{(\cdot)}^{(\bar{k})} \left(1 - \delta^{(\bar{k})}\right), \tag{4.37}$$

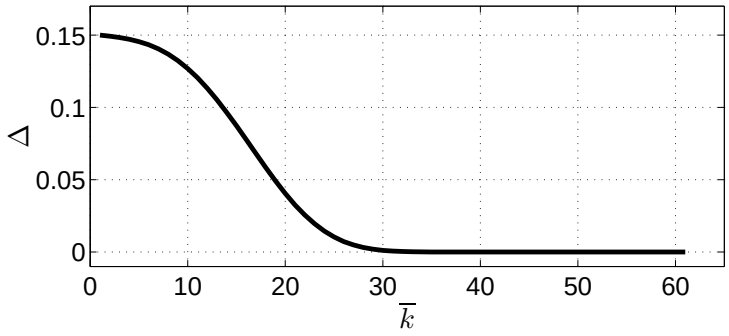


Fig. 4.4: Example of a decreasing tolerance with $\Delta^{(1)} = 0.15$, $N_\delta = 10^3$ and $\delta^{(1)} = 5 \cdot 10^{-3}$.

$$\delta^{(\bar{k}+1)} = \begin{cases} \delta^{(\bar{k})} + \dfrac{\bar{k}}{N_\delta} & \text{if } \delta^{(\bar{k})} + \dfrac{\bar{k}}{N_\delta} \leq 1\,, \\ 1 & \text{otherwise.} \end{cases} \tag{4.38}$$

where N_δ is a user-defined constant. Clearly, starting from $\Delta_{(\cdot)}^{(1)} \neq 0$, $\Delta_{(\cdot)}^{(\bar{k}+1)} = 0$ if and only if $\delta^{(\bar{k})} = 1$. Note that the condition in Eq. (4.38) is violated if:

$$\begin{aligned} \delta^{(\bar{k})} + \frac{\bar{k}}{N_\delta} &= \delta^{(\bar{k}-1)} + \frac{\bar{k}-1}{N_\delta} + \frac{\bar{k}}{N_\delta} \\ &= \delta^{(\bar{k}-2)} + \frac{\bar{k}-2}{N_\delta} + \frac{\bar{k}-1}{N_\delta} + \frac{\bar{k}}{N_\delta} \\ &= \ldots = \delta^{(1)} + \frac{1}{N_\delta} + \frac{2}{N_\delta} + \ldots + \frac{\bar{k}}{N_\delta} \\ &= \delta^{(1)} + \sum_{n=1}^{\bar{k}} \frac{n}{N_\delta} = \delta^{(1)} + \frac{1}{N_\delta}\frac{\bar{k}(\bar{k}-1)}{2} > 1 \end{aligned} \tag{4.39}$$

Eq. (4.39) is equivalent to:

$$\bar{k}^2 + \bar{k} - \left(2N_\delta\left(1 - \delta^{(1)}\right)\right) > 0, \tag{4.40}$$

which leads to the only acceptable solution $\bar{k}^* \in \mathbb{N}$:

$$\bar{k}^* = \left\lceil \frac{-1 + \sqrt{1 + 8N_\delta\left(1 - \delta^{(1)}\right)}}{2} \right\rceil. \tag{4.41}$$

In Eq. (4.41), $\lceil x \rceil$ is the *ceiling* function giving the smallest integer greater than or equal to x. As a consequence, with the decreasing tolerances modeled as in Eqs. (4.37)-(4.38), the two parameters $\delta^{(1)}$ and N_δ impose the initial slope of the curve in Fig. 4.4 and the number of external cycles $\bar{k}^*$ required to set the tolerances to zero.

4.5.3 Exit condition

With reference to what has been stated in Sec. 3.8.2 and introducing the user-defined tolerance ε_c, IPSO stops according to the following convergence criterion:

$$\delta\overline{J}_g = \frac{1}{M} \sum_{i=\hat{k}-M+1}^{\hat{k}} \frac{\overline{J}_g^{(i-1)} - \overline{J}_g^{(i)}}{\overline{J}_g^{(i)}} < \varepsilon_c, \tag{4.42}$$

where M is usually set equal to 10. Such an exit condition allows the PSO to converge until all the particles are within the same neighborhood around the detected minimum.

4.6 Global optimization

Choosing the PSO parameters such that a good balance between exploration and exploitation is guaranteed, the final solution *can* be the accepted as optimal solution. As it will be shown in the following chapters, Monte-Carlo simulation may help studying the convergence properties of the algorithm. For instance, one can understand if, solving several times the same optimization problem, the PSO always converges toward the same solution. However, we do not know if this solution is the best of all the possible solutions to the given problem. Indeed, in [27], when discussing about the optimality guaranteed by numerical softwares, it is stated that:

"*A practical test for optimality is not the claim of a globally optimal solution; rather, it is whether a new solution to an old problem is better than the old solution (and by how much) or whether the problem posed is itself new so that any (feasible) solution is desirable to the alternative (of no solution)*."

As a consequence, most of the time the optimality of the solution is stated basing on the personal experience and the knowledge of the problem reported in the literature by the scientific community.

4.7 Endnotes

In this chapter, the Inverse-dynamics Particle Swarm Optimization has been described. This method has been developed to solve optimal control problems by 1) using an inverse dynamics approach with differential flatness formulation and 2) solving the nonlinear parameter optimization problem with the Particle Swarm Optimization. These are the key points of the proposed algorithm, allowing the

technique to tackle optimal control problems using the minimum number of independent unknowns and searching for the global optimal solution by means of the swarm intelligence. Other important features have been described, such as the approximation of the unknown optimal functions with B-spline and the adaptive decreasing tolerances technique. All in all, the Inverse-dynamics Particle Swarm Optimization can guarantee the computation of feasible near-optimal solutions with the satisfaction of all the end-point equality constraints and the path inequality constraints.

Part III

Planning of Constrained Attitude Maneuvers

5

Time-optimal Reorientation Maneuvers with Keep-out Cones*

Abstract

The chapter deals with the problem of spacecraft time-optimal reorientation maneuvers under boundaries and path constraints. The minimum time solution with keep-out constraints is proposed using the Particle Swarm Optimization technique. The Inverse Dynamics Particle Swarm Optimization is used and compared with a direct dynamics approach. The comparison between basic and improved B-spline approximation is carried out. It is established that the computation of the minimum time maneuver with the proposed inverse technique leads to near optimal solutions, which fully satisfy all the boundaries and path constraints. Finally, the minimum-time planning is accomplished with a refined dynamical model including the reaction wheels dynamics.

Nomenclature

$\mathbf{x}$	=	PSO particle	$\boldsymbol{\mathcal{B}}$	=	Body (satellite) reference frame
$\mathcal{W}$	=	Satellite wheels	$\boldsymbol{I}$	=	Inertia tensor
$\boldsymbol{\omega}$	=	Body angular velocity	$\hat{(\cdot)}$	=	Unit vector
$\boldsymbol{u}$	=	External control	J	=	Performance index
t	=	Time (s)	$\mathcal{N}$	=	B-spline basis function
$(\cdot)^N$	=	Numerical approximation	$\mathcal{D}$	=	B-spline degree
$\mathbf{U}$	=	Approximation control point	$\dot{(\cdot)}$	=	First time derivative
$\ddot{(\cdot)}$	=	Second time derivative	$\boldsymbol{p}$	=	Modified Rodrigues Parameters
$\bar{k}$	=	External loop index	$(\cdot)_{0/f}$	=	Initial/final time value
$\boldsymbol{\sigma}$	=	Optical axis	$\boldsymbol{\sigma}_s$	=	Light-source direction
α_s	=	Light-source half-angle (rad)	β	=	Angle between $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}_s$ (rad)
$N_{\mathcal{P}}$	=	Number of approximation coefficients	$N_{\mathcal{S}}$	=	Number of PSO particles
$\boldsymbol{\eta}$	=	Quaternions	$\theta, \hat{\mathbf{n}}$	=	Angle-axis of rotation
Ξ, Ψ	=	Angular kinematics matrices	R	=	Rotation matrix
$\tilde{(\cdot)}$	=	Approximation parameter	u_{max}	=	Maximum external control
N_T	=	Number of discretization points	Δ	=	Penalty function tolerance
B	=	BC penalty function	P	=	PC penalty function
N_{viol}	=	Number of constraints violation	f	=	Feasibility penalty function

*This chapter is based on Refs. [1, 2, 3, 4].

$\mathscr{Y}_B$	=	Set of BC-compliant functions	$\mathbb{U}$	=	Control space
$\mathbb{Y}$	=	Flat output space	$\boldsymbol{\mathcal{B}}_0$	=	Inertial reference frame
Θ_f	=	Slew angle	$\mathcal{S}$	=	Satellite-fixed frame
$\mathcal{I}$	=	Inertial frame	$\boldsymbol{T}_{int}$	=	Internal torque
$\boldsymbol{H}$	=	Angular momentum			

5.1 Introduction

Time-optimal spacecraft reorientation maneuvers under boundaries and path constraints are difficult to compute, because the solutions are related to complex nonlinear problems. The minimum time reorientation maneuver of a rigid spacecraft is a well-known problem: the first related work regarding a numerical approach dates back to the 90s [80, 81, 82, 83].

Unconstrained, minimum-time rest-to-rest maneuvers through large angles (so-called slew maneuvers) were first taken into account by Bilimoria and Wie [80]. Considering a rigid spacecraft with spherically symmetric mass distribution and with equal control-torque authority for all three axes, they showed that the intuitively obvious rotation about the eigenaxis is not the time-optimal solution. Bai and Junkins [84] reconsidered this problem proving that if the total control vector is constrained to have a maximum magnitude (i.e., with the orthogonal control components not necessarily independent), then the time-optimal solution is the eigenaxis maneuver.

These papers have been used as a reference providing some test cases for successive works as in [85, 86, 87]; the research still focuses on this problem with several approaches, for example, through homotopic approach algorithms [88], pseudospectral optimization analysis [89, 90] or with hybrid numerical techniques [84].

The problem of reorientation maneuvers with path constraints has been initially studied by McInnes in [91]. The path constraints are determined by bright sources such as the Sun or the Moon that need to be avoided by payloads such as telescopes or sensors as with star trackers [92]. The path planning has to take into account exclusion cones with an angle greater than the effective angular diameter of the bright source, due to its very low apparent magnitude. Other path constraints could include pointing boundaries for antennas to maintain communication [93]. The angle of exclusion is determined by the sensor baffle and the source (considering its angular dimension and its intensity), and it is typically between 15 and 45 degrees depending on the sensitive optical sensor [94, 95, 96]. Several numerical methods, such as the Randomized Motion Planning [97], the Logarithmic Barrier Potentials

[98] or the Lie group variational integrator [99], have been used to obtain the optimal solution.

Heuristic and metaheuristic algorithms have been recently proposed for the planning of slew maneuvers, as in [100, 101] . Metaheuristic algorithms may be used to find sub-optimal solutions: for instance, Melton proposed a hybrid technique where a metaheuristic solution was used as the best available initial guess for a pseudospectral optimizer [102]. The hybrid technique has been newly proposed in [103], where the initial guess is carried out with the covariance matrix adaptation-evolutionary strategy. The metaheuristic algorithms are being studied extensively and the high interest generated from their results is shown in the research performed by NASA [104]. With regards to the PSO, it has been used not only for the planning of attitude maneuvers as in [105, 106], but also for the trajectory planning [107, 108, 109, 110, 111] or for attitude determination [112, 113].

From the previous works, the optimization of attitude maneuvers uses the PSO applied to the control. This approach here is referred to as Direct Method. This chapter shows how such an approach fails in satisfying the boundary constraints (i.e. final position and final velocity). In this following, the Inverse-dynamics Particle Swarm Optimization is applied to solve the reorientation problem, where the final boundary constraints are straight satisfied.

The chapter is organized as follows. In Sec. 5.2 the attitude reorientation problem is described. This problem will be taken into account also in Chapters 6-7. Sec. 5.3 and Sec. 5.4 describe the direct dynamics and the inverse dynamics approaches that have been investigated in Ref. [1]. The inverse-dynamics approach that will be described is very close to the final IPSO depicted in Chapter 4, even though the adaptive decreasing tolerances were not clearly designed and the basic B-spline approximation was used. In Sec. 5.6 the comparison between the direct and the inverse dynamics approaches is carried out. This section summarizes the results reported in Ref. [1], showing the better performances of the inverse method with regard to the direct method. In Sec. 5.7 the results of IPSO embedding the improved B-spline approximation are reported, showing improved performances with respect to those obtained with the basic B-spline approximation in Sec. 5.6.2. In Sec. 5.8 the reorientation problem is solved including in the satellite dynamical model the reaction-wheels dynamics. Here, the definitive version of the IPSO is employed. Concluding remarks are given in Sec. 5.9.

5.2 Problem statement

This study deals with the problem of a slew maneuver. A satellite must perform a reorientation maneuver where some optical instruments (for example a star tracker) cannot be exposed to sources of bright light such as the Earth, the Sun and the Moon. The maneuver angle Θ_f and the initial and final attitudes are known. The slewing motion must be constrained to prevent the sensor axis from entering into established "*keep-out*" zones known as cones. Such areas have central axes pointing to the Sun, the Earth and the Moon, and specified half-angles depending on the light magnitude, the distance from the satellite and the angular diameter of the source. Moreover, the maneuver must be *rest-to-rest*, i.e. the angular velocity must be equal to zero for $t = t_0$ and $t = t_f$. In particular t_0 is a constant parameter (that may be set equal to zero).

The objective is to minimize the maneuver overall time $t_f - t_0$, so the performance index is

$$J = t_f - t_0. \tag{5.1}$$

A definition for the used reference frames is needed in order to describe the maneuver. First, let us denote the body-fixed reference frame as $\boldsymbol{\mathcal{B}} = \{\hat{\boldsymbol{e}}_x, \hat{\boldsymbol{e}}_y, \hat{\boldsymbol{e}}_z\}$, whereas the body axes are $\boldsymbol{x_\mathcal{B}}$, $\boldsymbol{y_\mathcal{B}}$, $\boldsymbol{z_\mathcal{B}}$.

One important hypothesis is that the time for completing the maneuver is negligible with respect to the time for completing an orbit. In this case, with regards to an Earth-Centered Inertial reference frame ECI, it is possible to approximate the velocity of the satellite center of mass to zero. This approximation allows the definition of an inertial reference frame fixed in the original position of the body-fixed frame at time $t = t_0$ that will be referred to as $\boldsymbol{\mathcal{B}}_0 = \{\hat{\boldsymbol{e}}_{0,x}, \hat{\boldsymbol{e}}_{0,y}, \hat{\boldsymbol{e}}_{0,z}\}$. As a consequence, the positions of the keep-out cones defined in this inertial frame do not change during the maneuver. In the body-fixed frame, the body angular velocity is $\boldsymbol{\omega} = \omega_x\hat{\boldsymbol{e}}_x + \omega_y\hat{\boldsymbol{e}}_y + \omega_z\hat{\boldsymbol{e}}_z$ and the body torques are $\boldsymbol{u} = u_x\hat{\boldsymbol{e}}_x + u_y\hat{\boldsymbol{e}}_y + u_z\hat{\boldsymbol{e}}_z$.

The maneuver is a rotation around the initial body-fixed x-axis, i.e. $\hat{\boldsymbol{e}}_x$. In the case without path constraints, the minimum-time rotation takes place with nutation components around the y and the z axes, as known from the well-known results firstly presented in Ref. [80]. More details on the maneuvers without path constraints may be found in Chapter 6. If one or more keep-out cones are present, all the components of $\boldsymbol{\omega}$ are different from zero to guarantee a feasible maneuver and minimize the maneuver time by means of the nutation contributions. The rigid-body motion is described by the Euler's equation expressed in the most general form as

$$\boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \boldsymbol{u}, \tag{5.2}$$

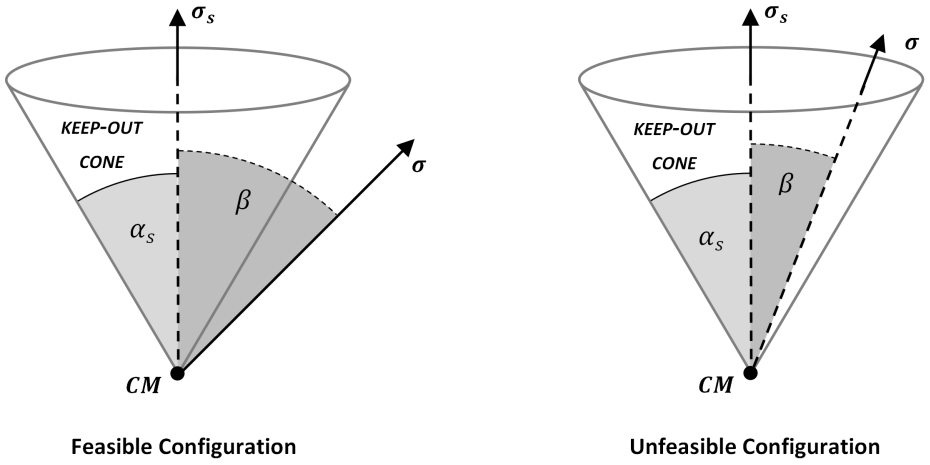


Fig. 5.1: Graphical representation of the keep-out cone constraint.

where $\boldsymbol{I}$ is the inertia tensor and $\boldsymbol{u}$ the total torque vector. In the following, three independent torques aligned with the axes of the body-fixed frame will be considered.

While moving from the initial to the final attitude, the sensor axis $\boldsymbol{\sigma}$ must be kept at least at the minimum angular distance α_s from each light source. Using the notation from [90] we can define the so-called *keep-out constraints* as

$$C_s(t) = \boldsymbol{\sigma}(t) \cdot \boldsymbol{\sigma}_s - \cos(\alpha_s) \leq 0 \qquad \forall t \in [t_0, t_f], \tag{5.3}$$

where $\boldsymbol{\sigma}(t)$ is the direction pointed to by the optical sensor and $\boldsymbol{\sigma}_s$ is vector placed in the center of mass of the satellite and pointing to the generic source of light, here represented by s.

The constraint is shown in Fig. 5.1. On the left, a feasible configuration is reported: the sensor axis $\boldsymbol{\sigma}$ is outside the keep-out cone. On the right, an unfeasible configuration is reported: in this case, the sensor axis $\boldsymbol{\sigma}$ is inside the keep-out cone. Introducing the new variable β defined as $\beta(t) = \cos^{-1}(\boldsymbol{\sigma}(t) \cdot \boldsymbol{\sigma}_s)$, the constraint in Eq. (5.3) may be re-written more easily as

$$\beta(t) \geq \alpha_s \qquad \forall t \in [t_0, t_f]. \tag{5.4}$$

The cone defined for each source of light will be referred to as *keep-out cone*.

5.3 Direct-dynamics approach

The first analysis has been made by directly integrating the control law, as in Ref. [100]. For clarification, Eq. (5.50) is used in the following form:

$$\dot{\boldsymbol{\omega}} = I^{-1}\left(\boldsymbol{u} - \boldsymbol{\omega} \times I\boldsymbol{\omega}\right) \tag{5.5}$$

This approach has been already described in Sec. 2.4.1. In the following pages the acronym DPSO will be used to refer to this technique, where "D" stands for *Direct* integration of the dynamics. This method is based on the approximation of the control policy and the numerical integration of the dynamics.

5.3.1 Description of the attitude kinematics

The attitude is described by quaternions of rotation given by

$$\bar{\boldsymbol{\eta}} = \begin{bmatrix} \boldsymbol{\eta} \\ \eta_4 \end{bmatrix} = \left[\eta_1 \;\; \eta_2 \;\; \eta_3 \;\; \eta_4\right]^T, \tag{5.6}$$

where

$$\boldsymbol{\eta} = \sin\left(\theta/2\right)\hat{\mathbf{n}}\,, \quad \eta_4 = \cos\left(\theta/2\right), \tag{5.7}$$

with $\hat{\mathbf{n}}$ representing the Euler rotation axis and θ is the Euler angle of rotation. The vector $\bar{\boldsymbol{\eta}}$ is a function of time, even though it is not explicitly reported.

The detailed description of the quaternions along with the description of the associated rotation matrix $R(\bar{\boldsymbol{\eta}})$ and the kinematic matrix $\Xi(\bar{\boldsymbol{\eta}})$ may be found in Ref. [114]. For completeness, the rotation matrix is

$$R(\bar{\boldsymbol{\eta}}) = \begin{bmatrix} \eta_1^2 - \eta_2^2 - \eta_3^2 + \eta_4^2 & 2(\eta_1\eta_2 + \eta_4\eta_3) & 2(\eta_1\eta_3 - \eta_4\eta_2) \\ 2(\eta_2\eta_1 - \eta_4\eta_3) & -\eta_1^2 + \eta_2^2 - \eta_3^2 + \eta_4^2 & 2(\eta_2\eta_3 + \eta_4\eta_1) \\ 2(\eta_3\eta_1 + \eta_4\eta_2) & 2(\eta_3\eta_2 - \eta_4\eta_1) & -\eta_1^2 - \eta_2^2 + \eta_3^2 + \eta_4^2 \end{bmatrix} \tag{5.8}$$

and the kinematic equation for the quaternions is

$$\dot{\bar{\boldsymbol{\eta}}} = \frac{1}{2}\,\Xi(\bar{\boldsymbol{\eta}})\,\boldsymbol{\omega}, \tag{5.9}$$

where the matrix $\Xi(\bar{\boldsymbol{\eta}})$ is defined as

$$\Xi(\bar{\boldsymbol{\eta}}) = \begin{bmatrix} \eta_4 & -\eta_3 & \eta_2 \\ \eta_3 & \eta_4 & -\eta_1 \\ -\eta_2 & \eta_1 & \eta_4 \\ -\eta_1 & -\eta_2 & -\eta_3 \end{bmatrix}. \tag{5.10}$$

Eq. (5.9) needs to be integrated with Eq. (5.5) in order to update the angular position.

The position of the rotated sensor axis $\boldsymbol{\sigma}(t)$ at some time instant t in the initial inertial reference frame $\boldsymbol{\mathcal{B}}_0$ is obtained rotating the initial position of sensor axis $\boldsymbol{\sigma}(t_0)$ as

$$\boldsymbol{\sigma}(t) = R(\bar{\boldsymbol{\eta}})^T \, \boldsymbol{\sigma}(t_0). \tag{5.11}$$

5.3.2 Definition of the particle

Each particle of the swarm contains information about the control law in the body-axes and the final time for the maneuver. The ith particle is an array containing:

- The maneuver time t_f.

- The coefficients for the polynomial approximation of the control policy. The external control $\boldsymbol{u}(t)$ is approximated with a cubic spline function $\boldsymbol{u}^N(t)$ defined by the spline control points $\mathbf{U}_{j,k} = [t_k, \tilde{u}_{j,k}]$, where $j = 1, 2, 3$ is the maneuver axis and $k = 1, 2, ..., N_{\mathcal{P}}$, being $N_{\mathcal{P}}$ the number of points that will be used for the interpolation of the control. The $N_{\mathcal{P}}$ points are associated to time instants equally spaced between t_0 and t_f. The approximated control $\boldsymbol{u}^N(t)$ is discretized in $N_T + 1$ integration points (so that $t = t_0, \ t_1, \ ..., \ t_{N_T} = t_f$) interpolating the $N_{\mathcal{P}}$ points $\mathbf{U}_{j,k}$ with cubic splines. Using the PSO displacement rules, each control policy generated with the spline approximation is feasible since

  $$\text{if } |u_j^N(t_i)| > u_{max} \Rightarrow |u_j^N(t_i)| = u_{max}, \quad \forall \, 0 \leq i \leq N_T, \ j = 1, 2, 3. \tag{5.12}$$

  In this way, the control path constraint is imposed a-priori and does not need to be taken into account in the optimization process.

To summarize, the DPSO particle is defined as

$$\mathbf{x} = [\tilde{u}_{1,1}, ..., \tilde{u}_{1,N_{\mathcal{P}}}, \tilde{u}_{2,1}, ..., \tilde{u}_{2,N_{\mathcal{P}}}, \tilde{u}_{3,1}, ..., \tilde{u}_{3,N_{\mathcal{P}}}, t_f], \tag{5.13}$$

or, by defining $\tilde{\boldsymbol{u}}_i = [\tilde{u}_{i,1}, ..., \tilde{u}_{i,N_{\mathcal{P}}}]$,

$$\mathbf{x} = [\tilde{\boldsymbol{u}}_1, \tilde{\boldsymbol{u}}_2, \tilde{\boldsymbol{u}}_3, t_f]. \tag{5.14}$$

With regard to the PSO velocity constraints, $\Delta\tilde{u}_{j,k}$ and Δt_f are defined as the PSO velocities associated with the control and the maneuver time of each particle. Eq. (3.9) is taken into account as

$$\begin{gathered} |\Delta\tilde{u}_{j,k}| \leq 0.2 \cdot u_{max} \ , \ \ |\tilde{u}_{j,k}| \leq u_{max}, \\ |\Delta t_f| < 0.1 \cdot (t_{max} - t_{min}) \ , \ \ t_{min} < t_f < t_{max}, \\ k = 1, ..., N_{\mathcal{P}} \ , \ \ j = 1, 2, 3. \end{gathered} \tag{5.15}$$

where u_{max} is the known maximum value of the actuators, while t_{max} and t_{min} may be defined by knowing the unconstrained solution. The values 0.1 and 0.2 are chosen in order to make the maximum velocities equal to the 10% of the dynamic range of the particles, as already explained in relation to Eq. (3.9).

The initialization of the time and the control of each particle is based on a uniform random distribution of the particles within the constraints of Eq. (5.15).

5.3.3 Implementation of the DPSO

Recalling what has been stated in Sec. 2.4.1, let us summarize the details of the optimization problem so far outlined. Let us write the optimization problem as a function of the approximated state and control, i.e. $\boldsymbol{\eta}^N$, $\boldsymbol{\omega}^N$ and $\boldsymbol{u}^N$. It is noteworthy to remind the reader that we approximate only the control and derive the angular acceleration and the attitude by integrating the rotational dynamics and the kinematics. The optimization of the constrained slew maneuver with the direct dynamics approach has the following mathematical form:

$$\begin{aligned} &\text{Find } \boldsymbol{u}^N(t) : t \to \mathbb{U} \subset \mathbb{R}^{N_u},\ t_f \in \mathbb{R} \\ &\text{minimizing} \\ &J^N = t_f - t_0 \\ &\text{subject to, } \forall t \in [t_0, t_f] \\ \text{Dynamic constraints:} \quad &\dot{\boldsymbol{\omega}}^N = \boldsymbol{I}^{-1} \left(\boldsymbol{u}^N - \boldsymbol{\omega}^N \times \boldsymbol{I}\boldsymbol{\omega}^N \right) \\ &\dot{\bar{\boldsymbol{\eta}}}^N = \frac{1}{2} \, \Xi(\bar{\boldsymbol{\eta}}^N) \, \boldsymbol{\omega}^N \\ \text{Boundary constraints:} \quad &\boldsymbol{\omega}^N(t_0) = \mathbf{0} \quad \boldsymbol{\omega}^N(t_f) = \mathbf{0} \\ &\bar{\boldsymbol{\eta}}^N(t_0) - \bar{\boldsymbol{\eta}}_0 = \mathbf{0}, \quad \bar{\boldsymbol{\eta}}^N(t_f) - \bar{\boldsymbol{\eta}}_f = \mathbf{0}, \\ \text{State path constraints:} \quad &R(\bar{\boldsymbol{\eta}}^N)^T \, \boldsymbol{\sigma}(t_0) \cdot \boldsymbol{\sigma}_s - \cos\alpha_s \leq 0. \end{aligned} \tag{5.16}$$

In Eq. (5.16), $\bar{\boldsymbol{\eta}}_0$ and $\bar{\boldsymbol{\eta}}_f$ denote the exact initial and final angular positions; the angular velocity must be zero in $t = t_0$ and in $t = t_f$. The end-point constraints

are equality constraints, while the path constraint is in the form of an inequality constraint. The solutions that satisfy all the boundary and path constraints lie inside the FSS. Note that there is no control constraint, as every solution considered by the optimizer does automatically satisfy $\|\boldsymbol{u}(t)\|_\infty - u_{max} \leq \mathbf{0}$ (as usual the L_∞-norm is defined as $\|\boldsymbol{u}(t)\|_\infty = \max\{|u_i(t)| : i = 1, 2, 3\}$). The dynamics constraint is satisfied via numerical integration.

The swarm progressively moves towards the individuals (if the *local best* search is chosen) or individual (if instead the *global best* search is preferred), which provides the minimum time of maneuver, ensuring the fulfillment of boundary and path constraints.

The fitness function is selected in the form of an *Exterior Penalty Function* (explained in Sec. 1.3.2). However, differently from the inverse dynamics approach described in Sec. 4.5 where only inequality path constraints were considered, the direct approach requires the introduction of penalty functions for equality constraints. Accordingly, the extended performance index is

$$\overline{J}^N = t_f + b\sum_{i}^{N_{eq}} B_i + \pi P + \mu\, N_{viol} + f \tag{5.17}$$

where B_i is the penalty function for the i^{th} boundary equality constraint with weight b_i, P is the penalty function for the path inequality constraints with weight π, N_{viol} is the number of violated constraints with weight μ and f is a feasibility parameter. The equality constraints (i.e the boundary conditions in Eq. (5.16)) have been divided into position and velocity.

Given the end-point constraints in Eq. (5.16), $N_{eq} = 2$ and the penalty functions B_1, B_2 are

$$\begin{aligned} B_1 &= \sum_{j=1}^{3} \max\{0, \left|\omega_j^N(t_f)\right| - \Delta_{eq_1}\}, \\ B_2 &= \sum_{j=1}^{4} \max\{0, \left|\eta_j^N(t_f) - \eta_{j,f}\right| - \Delta_{eq_2}\}. \end{aligned} \tag{5.18}$$

The constraints are taken into account with decreasing tolerances and Δ_{eq_1} and Δ_{eq_2} tend to zero according to the basic decreasing law given in Sec. 4.5.2. However, equality tolerances are never set to zero, as the optimizer is not able to exactly satisfy equality constraints. This issue is related to the numerical approximation of the control and the numerical errors introduced by the numerical integrator.

The path penalty function P, defined as in Sec. 4.5, is

$$P = \left(\sum_{j=0}^{N_T} s_i(t_j) + \sum_{j=0}^{N_T} m_i(t_j) \right), \tag{5.19}$$

where $s_i(t_i)$ and $m_i(t_i)$ are

$$s_i(t_j) = \begin{cases} 0 & \text{if } \boldsymbol{\sigma}^N(t_j) \cdot \boldsymbol{\sigma}_S - \cos\alpha_S < \Delta_S \\ 1 & \text{otherwise} \end{cases} \tag{5.20}$$

$$m_i(t_j) = \begin{cases} 0 & \text{if } \boldsymbol{\sigma}^N(t_j) \cdot \boldsymbol{\sigma}_M - \cos\alpha_M < \Delta_M \\ 1 & \text{otherwise} \end{cases} \tag{5.21}$$

and Δ_S and Δ_M are the constraint tolerances. The angles α_S and α_M are the exclusion angles of the Sun and the Moon, respectively. The inequality constraints are fully satisfied only when $\Delta_S = \Delta_M = 0$. Similarly to the case of the equality constraints, Δ_S and Δ_M are decreasing tolerances following the basic law of Sec. 4.5.2.

The term N_{viol} is the counter that considers the violated constraints. Every time the global best particle reaches the value of $N_{viol} = 0$ the precision is improved, and the tolerances decrease. The control constraint in Eq. (5.16) is accounted for in Eq. (5.15). As described in Sec. 4.5.1, we can recognize one internal and one external loop. In the internal loop, the swarm tries to find a feasible minimum-time solution considering the current value of the tolerances. As soon as a feasible solution has been found, the tolerances values are updated in the external loop.

The last parameter f in Eq. (5.17) may be regarded as a *feasibility* constraint and it takes into account the geometry of the problem. This term is set to 0 if the keep-out cones do not intersect. In this case a generic maneuver can pass through the keep-out cones, even though the best maneuver may lie outside this region. On the other hand, if the keep-out cones intersect, then $f = f_{max}$ if the optical sensor axis $\boldsymbol{\sigma}$ passes between the axes of the cones and $f = 0$ otherwise.

Algorithm 1 reports the main characteristics of the DPSO algorithm. Denoting with $\bar{k}$ the index of the external loop, the constraint tolerances decrease according to the following scheme:

1. For $\bar{k} = 1$ the keep-out cones constraint is not considered. The swarm moves towards the maneuver that goes from the initial to the final points, thus minimizing the time. If the optimal maneuver for this unconstrained case is known, it can be used as a guess: in this case, only one particle of the swarm is set to take the form of this optimal maneuver. In this manner, all the other

Algorithm 1: DPSO algorithm

```
Initialization of constants, swarm and tolerances;
while δJ̄_g > ε_c do
    update constraints tolerances;
    reset J_{p,i}^(k), J_g^(k), J_{l,i}^(k) ∀i = 1, ..., N_S;
    while N_viol^(g_best) > 0 and δJ̄_g > ε_c do
        update w, c_l, c_g;
        for i = 1 : N_S do
            evaluate the control approximation;
            integrate Euler's Equation and the kinematics;
            compute the sensor orientation;
            compute the extended performance index;
            update J_{p,i}^(k), J_g^(k), J_{l,i}^(k);
            update N_viol^(g_best)
        end
        for i = 1 : N_S do
            update the i-th particle velocity;
            update the i-th particle position;
        end
    end
end
```

particles move towards this position, ensuring that the swarm starts the search inside the FSS.

2. From $\bar{k} = 2$ to $\bar{k} = \bar{k}^*$ (where $\bar{k}^*$ may be defined by the user) the keep-out cones constraint is gradually included. Consequently, if all constraints are satisfied, the tolerances decrease as j increases from $\bar{k} = 2$ to $\bar{k} = \bar{k}^*$. In particular, in this first part of the loop the local search is set in order to be more significant than the global search. As a result, the swarm is divided into different search groups that increase the probability to find the global minimum of the problem.

3. From $\bar{k} = \bar{k}^*+1$ until the end of the external loop, the keep-out cones constraint is entirely included (i.e. $\Delta_{Sun} = \Delta_{Moon} = 0$), and the swarm will continue to minimize the maneuver time, improving the accuracy of the solution. Note that Δ_{eq_1} and Δ_{eq_2} are never set equal to zero.

The exit criterion is the one described in Sec. 4.5.3 with $M = 3$, i.e.

$$\delta\overline{J}_g = \frac{1}{3}\sum_{i=\hat{k}-2}^{\hat{k}} \frac{\overline{J}_g^{(i-1)} - \overline{J}_g^{(i)}}{\overline{J}_g^{(i)}} < \varepsilon_c, \tag{5.22}$$

where the superscript N has not been reported and $\hat{k}$ is the iteration index where the global best particle is updated.

5.4 Inverse-dynamics approach

The direct integration of the control law is a simple and traditional way of dealing with these types of problems. However, it shows some issues due to the low speed of the integration process and because it is subject to numerical errors. In fact, in the numerical code, the use of a function for the integration of the Euler equations is necessary and it affects the computational performances when high performances are required. Moreover, boundary constraints cannot be completely satisfied.

In this section the IPSO is used to obtain the solution to the minimum-time reorientation problem. The novelties and the main important features of the IPSO method have been already described in Chapter 4. In this section the very first version the IPSO is described (refer to Ref. [1]). The most important differences with the definitive IPSO version described in Chapter 4 are 1) the basic B-spline approximation is employed and 2) the basic adaptive decreasing tolerances consisting of a piecewise linear law (described in Sec. (4.5.2)) is adopted.

As it will be described, the IPSO may be used as a sub-optimal planner or in combination with a Pseudospectral Optimal Control Software (POCS) (in this work, the GPOPS software is employed, see Ref. [57]) to provide a feasible near-optimal initial guess.

5.4.1 Attitude parametrization

According to the theory introduced in Chapter 4, the inverse dynamics approach is based on the identification of a flat output vector, $\boldsymbol{y}$, with the same dimension of the control. Accordingly, the state must be found as a closed-form analytical function of the flat output as well as the control, which is evaluated by means of the Euler's equation (5.50). As a consequence, $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$ must be expressed in function of $\boldsymbol{y}$ and its time derivatives in order to obtain $\boldsymbol{u}$.

In this section it is proved that the Modified Rodriguez Parameters vector may be chosen as flat output. Since the control is defined in $\mathbb{R}^3$, an attitude description with only three parameters is required to use the differential flatness formulation. The Modified Rodrigues Parameters (MRPs, see Ref. [114] for further details) are chosen as they show no singularity during the maneuver when Θ_f is below 2π.

The mathematical formulation that describes the kinematics through the MRPs is reported below. A vector $\boldsymbol{p}$ is defined as

$$\boldsymbol{p} = \frac{\boldsymbol{\eta}}{1+\eta_4}, \tag{5.23}$$

where $\boldsymbol{\eta}$ and η_4 are the vectorial and scalar components of the quaternion. Moreover $\boldsymbol{p}$ may be rewritten in terms of axis and angle of rotation as

$$\boldsymbol{p}(\hat{\mathbf{n}}, \theta) = \tan(\theta/4)\hat{\mathbf{n}}. \tag{5.24}$$

The rotation matrix using MRPs is

$$R(\boldsymbol{p}) = I + \frac{4\left(1 - |\boldsymbol{p}|^2\right)}{(1 + |\boldsymbol{p}|^2)^2}\left[\tilde{\boldsymbol{p}}\right] + \frac{8}{(1 + |\boldsymbol{p}|^2)^2}\left[\tilde{\boldsymbol{p}}\right]^2, \tag{5.25}$$

where $[\tilde{\boldsymbol{p}}]$ is defined as

$$\left[\tilde{\boldsymbol{p}}\right] = \begin{bmatrix} 0 & p_3 & -p_2 \\ -p_3 & 0 & p_1 \\ p_2 & -p_1 & 0 \end{bmatrix}. \tag{5.26}$$

The axis angle at time t is given by

$$\boldsymbol{\sigma}(t) = R(\boldsymbol{p})^T \boldsymbol{\sigma}(t_0). \tag{5.27}$$

The derivative of the MRPs vector is related to the angular velocity by the equation

$$\dot{\boldsymbol{p}} = \frac{1}{4}\Psi(\boldsymbol{p})\boldsymbol{\omega}, \tag{5.28}$$

where the matrix $\Psi(\boldsymbol{p})$ is defined as

$$\Psi(\boldsymbol{p}) = \left[\left(1 - \boldsymbol{p}^T\boldsymbol{p}\right) I + 2[\tilde{\boldsymbol{p}}] + 2\boldsymbol{p}\boldsymbol{p}^T\right]. \tag{5.29}$$

For the following development of the inverse dynamics with PSO algorithm, it is necessary to find $\boldsymbol{\omega}(\boldsymbol{p}, \dot{\boldsymbol{p}})$ and $\dot{\boldsymbol{\omega}}(\boldsymbol{p}, \dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}})$. As far as the former vector is concerned, it is quite simple to obtain it from Eq. (5.28) as

$$\boldsymbol{\omega} = 4\Psi^{-1}(\boldsymbol{p})\dot{\boldsymbol{p}}. \tag{5.30}$$

Imposing the MRPs vector as flat output, i.e. $\boldsymbol{y} = \boldsymbol{p}$, Eq. (4.1) can be written for the slew maneuver problem as

$$\boldsymbol{x} = \boldsymbol{a}\left(\boldsymbol{p}, \dot{\boldsymbol{p}}\right) = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} I_{3\times 3} \\ 4\Psi^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ \dot{\boldsymbol{p}} \end{bmatrix}. \tag{5.31}$$

The parameter β in Eq. (4.1) is clearly equal to 1. The matrix Ψ^{-1} is defined as a near-orthogonal matrix since its inverse matrix is proportional to its transpose, i.e.

$$\Psi^{-1}(\boldsymbol{p}) = \frac{\Psi^T(\boldsymbol{p})}{(1 + \boldsymbol{p}^T\boldsymbol{p})^2}. \tag{5.32}$$

From Eq. (5.30) an important consequence may be drawn, that is

$$\boldsymbol{\omega} = \mathbf{0} \iff \dot{\boldsymbol{p}} = \mathbf{0}, \tag{5.33}$$

meaning that a zero angular velocity is obtained whenever the time derivative of the MRPs vector is zero. From Eq. (5.30), $\dot{\boldsymbol{\omega}}(\boldsymbol{p}, \dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}})$ may be derived as

$$\dot{\boldsymbol{\omega}} = 4\left(\dot{\Psi}^{-1}(\boldsymbol{p})\,\dot{\boldsymbol{p}} + \Psi^{-1}(\boldsymbol{p})\,\ddot{\boldsymbol{p}}\right), \tag{5.34}$$

where $\dot{\Psi}$ and $\dot{\Psi}^{-1}$ are evaluated as

$$\dot{\Psi} = \left[-(\dot{\boldsymbol{p}}^T\boldsymbol{p} + \boldsymbol{p}^T\dot{\boldsymbol{p}})I + 2[\tilde{\dot{\boldsymbol{p}}}] + 2(\dot{\boldsymbol{p}}\boldsymbol{p}^T + \boldsymbol{p}\dot{\boldsymbol{p}}^T)\right], \tag{5.35}$$

$$\dot{\Psi}^{-1} = \frac{\dot{\Psi}^T}{(1+\boldsymbol{p}^T\boldsymbol{p})^2} - \frac{2\Psi^T}{(1+\boldsymbol{p}^T\boldsymbol{p})^3}(\dot{\boldsymbol{p}}^T\boldsymbol{p} + \boldsymbol{p}^T\dot{\boldsymbol{p}}). \tag{5.36}$$

These equations fully describe the attitude kinematics through the MRPs vector. The main feature of the above equations is that an analytical closed-form solution is found to compute $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$. Although the mathematical form of these equations is more complex than the mathematical form described in the attitude kinematics with the quaternions, the advantage is in dealing with square matrices. In order to summarize these results, placing Eq. (5.30) and (5.34) in Eq. (5.50), we can identify $\boldsymbol{b}$ in Eq. (4.1) as the nonlinear function

$$\boldsymbol{u} = \boldsymbol{b}(\boldsymbol{p}, \dot{\boldsymbol{p}}, \ddot{\boldsymbol{p}}) = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}. \tag{5.37}$$

5.4.2 Definition of the particle

As previously stated in this section, the original method presented in Ref. [1] was based on the basic B-spline approximation. Hence, similarly to the previous direct-dynamics case, the particles are arrays containing:

- The value of the maneuver time t_f.
- The angular displacement parameters $\tilde{p}_{j,k}$ with $j = 1, 2, 3$ and $k = 1, 2, ..., N_{\mathcal{P}}$. As before, $N_{\mathcal{P}}$ is the number of points used for the interpolation of the angular displacement. The $N_{\mathcal{P}}$ points are associated to time instants equally spaced between t_0 and t_f. The kinematics is obtained in $N_T + 1$ points (so that $t = t_0,\ t_1,\ ...,\ t_{N_T} = t_f$) interpolating the $N_{\mathcal{P}}$ points $\mathbf{U}_{j,k} = [t_k, \tilde{p}_{j,k}]$ with B-splines.

To summarize, the IPSO particle is defined as:

$$\mathbf{x} = [\tilde{p}_{1,1}, ..., \tilde{p}_{1,N_{\mathcal{P}}}, \tilde{p}_{2,1}, ..., \tilde{p}_{2,N_{\mathcal{P}}}, \tilde{p}_{3,1}, ..., \tilde{p}_{3,N_{\mathcal{P}}}, t_f], \tag{5.38}$$

or, by defining $\tilde{\boldsymbol{p}}_i = [\tilde{p}_{i,1}, ..., \tilde{p}_{i,N_\mathcal{P}}]$,

$$\mathbf{x} = [\tilde{\boldsymbol{p}}_1, \tilde{\boldsymbol{p}}_2, \tilde{\boldsymbol{p}}_3, t_f]. \tag{5.39}$$

As in the previous approach, $\Delta\tilde{p}_j(k)$ and Δt_f are the velocities associated to the kinematics and the maneuver time of each particle of the swarm. In this case Eq. (3.9) takes the following form:

$$\begin{gathered} |\Delta\tilde{p}_{j,k}| \leq 0.2 \cdot \tan(\theta^*/4) \ , \ \ |\tilde{p}_{j,k}| \leq \tan(\theta^*/4) \\ |\Delta t_f| < 0.1 \cdot (t_{max} - t_{min}) \ , \ \ t_{min} < t_f < t_{max} \\ k = 1, ..., N_\mathcal{P} \ , \ \ j = 1, 2, 3. \end{gathered} \tag{5.40}$$

The expression $\tan(\theta^*/4)$ is explained in Eq. (5.24), while θ^* is an angle which satisfies $\theta^* \geq \Theta_f$, being Θ_f the imposed angle of maneuver. The time constraints t_{max} and t_{min} may be defined by knowing the unconstrained solution. The values 0.1 and 0.2 are selected to make the maximum velocities equal to the 10% of the dynamic range of the particles, as already explained with regards to Eq. (3.9).

The initialization of the kinematics and the maneuver time of each particle is based on a uniform random distribution of the particles within the constraints of Eq. (5.40).

5.4.3 Implementation of the basic IPSO

Considering that boundary constraints are a-priori satisfied with the technique presented in Sec. 4.3, the inverse dynamics formulation of the problem is summarized as

$$\begin{gathered} \text{Find } \boldsymbol{p}^N(t) : t \to \mathbb{Y} = \mathbb{R}^{N_u} \text{ such that } \boldsymbol{p}^N \in \mathscr{Y}_B,\ t_f \in \mathbb{R} \\ \text{minimizing} \\ J^N = t_f - t_0 \\ \text{subject to, } \forall t \in [t_0, t_f] \\ \text{State path constraint:} \quad R(\boldsymbol{p}^N)^T\, \boldsymbol{\sigma}(t_0) \cdot \boldsymbol{\sigma}_s - \cos\alpha_s \leq 0, \\ \text{Control path constraint:} \quad \|\boldsymbol{b}(\boldsymbol{p}^N(t), \dot{\boldsymbol{p}}^N(t), \ddot{\boldsymbol{p}}^N(t))\|_\infty - u_{max} \leq \mathbf{0} \end{gathered} \tag{5.41}$$

As in Sec. 4.3, $\mathscr{Y}_B$ is the set of the flat output approximation functions that satisfy the boundary constraints. The extreme simplification of this approach lies in the fact that both the boundary constraints and the initial conditions are imposed *a priori* for

each particle. Moreover, the dynamic constraints is no more reported as it is satisfied analytically by means of Eq. (5.37).

The fitness function is chosen as in Sec. 4.5, i.e.

$$J^N = t_f + \pi P + \gamma C + \mu\, N_{viol} + f \tag{5.42}$$

where π, γ and μ are user-defined weights, P is defined as in Eq. (8.36) and the control penalty function C is

$$C = \sum_{i=1}^{3} \sum_{j=0}^{N_t} c_i(t_j) \tag{5.43}$$

where $c_i(t_i)$ is

$$c_i(t_j) = \begin{cases} 0 & \text{if } u_i(\boldsymbol{p}(t), \dot{\boldsymbol{p}}(t), \ddot{\boldsymbol{p}}(t)) < u_{max} \\ 1 & \text{otherwise} \end{cases} \tag{5.44}$$

This term penalizes the particles whose values of the control exceed u_{max}. The meanings of N_{viol}, G_i and f is the same as in Eq. (5.17). As it can be seen, the fitness function does not need to consider the equality constraints as each particle is built to fully satisfy the boundary constraints. Consequently, no decreasing tolerances are introduced for the final attitude and angular velocity.

To clarify the main concepts of the proposed approach, the algorithm is reported in Algorithm 2. For completeness, the phases in which the algorithm is subdivided are reported. Though they are quite similar to those showed in the previous section, these phases also underline the fundamental differences between the IPSO and the DPSO. Let us call with $\bar{k}$ the index of the external cycle, where the tolerances values are updated. Hence, the tolerances decrease according to the following scheme:

1. For $\bar{k} = 1$ the keep-out cones constraint is not considered. The swarm must fly towards the maneuver from the initial point to the final point minimizing the time. If the optimal maneuver is known, it can be used as the initial guess: in this case, only one particle of the swarm takes the form of this optimal maneuver. Differently from the DPSO, in this case the maneuver starts from the exact point and ends at the exact point. Particles with values of the control that exceed M_{max} are penalized in the fitness function.

2. From $\bar{k} = 2$ to $\bar{k} = \bar{k}^*$ the keep-out cones constraint is gradually included, i.e. it is included with a user-defined tolerance. This tolerance decreases according to user-defined piecewise linear law only when the actual global best solution of the swarm satisfies the keep-out cones constraint. Consequently, as $\bar{k}$ increases

from $\bar{k} = 2$ to $\bar{k} = \bar{k}^*$, if the keep-out cones constraint is satisfied, the tolerance is decreased. In particular, in order to avoid the risk of converging to a local minimum, in the first cycle the local version of PSO is adopted instead of the global version.

3. From $\bar{k} = \bar{k}^* + 1$ to the end of the optimization, the keep-out cones constraint is totally included, and the swarm will continue to minimize only the maneuver time. Instead of optimizing final position, final velocity and time as in DPSO, in this case only the time must be optimized, being final position and final velocity set to their precise value *a priori* as mentioned above.

The esit criterion is defined as in Eq. (5.22).

5.5 Definition of the test case

In this section the general test case used from this point until the end of the present chapter is outlined. A satellite for Earth observation in LEO is taken as test case. The nominal attitude is defined as:

- The $\boldsymbol{z_{\mathcal{B}}}$ axis points in the nadir direction towards the Earth.

- The $\boldsymbol{x_{\mathcal{B}}}$ axis is in the direction of the spacecraft velocity vector for circular orbits.

Algorithm 2: IPSO algorithm

```
Initialization of constants, swarm and tolerances;
while δ̄J_g > ε_c do
    update constraints tolerances;
    reset J_{p,i}^(k), J_g^(k), J_{l,i}^(k) ∀i = 1, ..., N_S;
    while N_viol^(g_best) > 0 and δ̄J_g > ε_c do
        update w, c_l and c_g;
        for i = 1 : N_S do
            approximate p and evaluate ṗ, p̈;
            compute the sensor orientation;
            compute of ω, ω̇ and u;
            compute the extended performance index;
            update J_{p,i}^(k), J_g^(k), J_{l,i}^(k);
            update N_viol^(g_best)
        end
        for i = 1 : N_S do
            update the i-th particle velocity;
            update the i-th particle position;
        end
    end
end
```

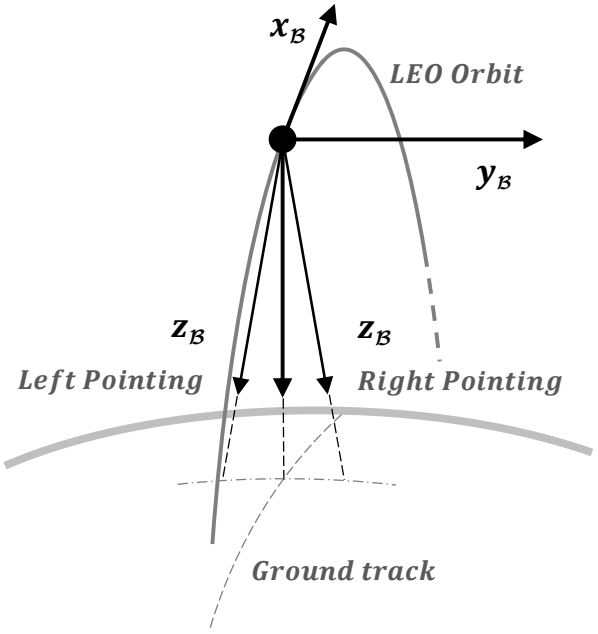


Fig. 5.2: Right and left looking configurations of the Body Reference Frame.

- The $y_{\mathcal{B}}$ axis completes the right-handed coordinate system and it is perpendicular to the orbital plane in the negative orbit normal direction.

A typical operative maneuver is defined as a rotation around the $x_{\mathcal{B}}$ axis (*roll rotation*): the satellite switches between the *Right-Pointing*, with a positive roll angle and $z_{\mathcal{B}}$ and pointing to the right with respect to the ground track, and the symmetric *Left-Pointing* configuration with a negative roll angle, see Fig. 5.2. In the following simulation roll angles set to $\Theta_f = 60°$ and $\Theta_f = 135°$ will be used to switch from the Right-Pointing position to the Left-Pointing position.

Assuming that the inertia tensor is diagonal in the $\mathcal{B}$ and has the following values:

$$I = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix} = \begin{bmatrix} 3000 & 0 & 0 \\ 0 & 4500 & 0 \\ 0 & 0 & 6000 \end{bmatrix} kg \cdot m^2 \tag{5.45}$$

Let us consider a star tracker sensor mounted on the $y_{\mathcal{B}} z_{\mathcal{B}}$ plane with the unit vector $\boldsymbol{\sigma}$ expressed in the in BRF as:

$$\boldsymbol{\sigma} = 0\,\hat{\boldsymbol{e}}_1 - 0.62\,\hat{\boldsymbol{e}}_2 - 0.79\,\hat{\boldsymbol{e}}_3 \tag{5.46}$$

We assume that the attitude maneuvers are obtained through three independent torques aligned with the axes of the body reference frame $\mathcal{B}$ with same maximum value $u_{max} = 0.25$ N-m.

Two keep-out cones are considered, centered respectively on the Sun and on the Moon. The corresponding half-angles are set to 40 deg and 17 deg. The orientations of the keep-out cones are reported in the following section.

5.6 Comparison of DPSO and basic IPSO

Three different case studies are proposed, whose characteristics are reported in the Tables 5.2. The roll angle is set to $\Theta_f = 60$ deg for cases 1 and 2, whereas $\Theta_f = 135$ deg for the third case. The free angle is the space between the two keep-out cones. As a further explanation, the geometry of scenario 1 is represented in Fig. 5.3, where the two keep-out cones, the optimal maneuver and the inertial frame are reported.

The particles explore the search space by means of the unified PSO velocity expressed in Eq. (3.12). The inertia weight w, the local best constant c_l and the global best constant c_g decrease according to Eq. (3.8) and Eq. (3.13). The value of the parameter K is reported in Table 5.3-5.4, along with all the PSO parameters, for both DPSO and IPSO. The iteration index j^* related to the setting $\Delta_{ineq} = 0$ is the external iteration related to $\Delta = 0.08$. The total number of external iterations is a function of the tolerance criteria: both the DPSO and the IPSO stop according to the criterion in Eq. (4.42) with $\varepsilon_c = 10^{-8}$ and $M = 3$.

For numerical reasons, normalized units are considered: the control is divided by u_{max}, the inertia matrix by I_1 and the maneuver time by $\sqrt{I_1/u_{max}}$. As a consequence, Eq. (5.17) and Eq. (5.63) are non- dimensional.

For scenario 1, the PSO solutions will be shown and compared with the optimal solution obtained with a Pseudospectral Optimal Control Software POCS (the free version of GPOPS-II for academic purposes has been used, see Ref. [57] for details).

Tab. 5.2: Direction of Sun and Moon in $\mathcal{B}_0$ for the proposed scenarios.

Scenario			Sun (in $\mathcal{B}_0$)			Moon (in $\mathcal{B}_0$)			Free Angle
N.	Θ_f (deg)	type	$\boldsymbol{\sigma}_{S,x}$	$\boldsymbol{\sigma}_{S,y}$	$\boldsymbol{\sigma}_{S,z}$	$\boldsymbol{\sigma}_{M,x}$	$\boldsymbol{\sigma}_{M,y}$	$\boldsymbol{\sigma}_{M,z}$	(deg)
1	60	roll	-0.58	-0.08	-0.81	0.41	-0.13	-0.91	0.52
2	60	roll	-0.65	0.28	-0.71	0.15	-0.25	-0.96	0.84
3	135	roll	-0.18	0.56	0.81	0.95	0.00	0.31	26.41

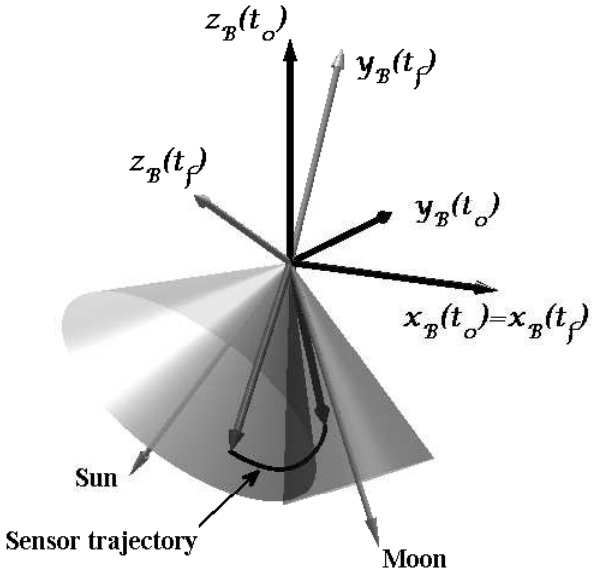


Fig. 5.3: 3D plot of the basic IPSO solution for scenario 1.

Tab. 5.3: Basic IPSO constant parameters

Parameter	Value	Parameter	Value
$N_{\mathcal{S}}$	30	ε_c	10^{-8}
$N_{\mathcal{P}}$	8	b	10
$\mathcal{D}$	7	π	10
K	1000	γ	10
c_p	1.5	μ	10
f_{max}	10^{10}	L_R	3

Tab. 5.4: Basic IPSO variable parameters

Parameter	Initial Value	Final Value
$\Delta_{eq_{(\cdot)}}$	0.14	*variable*
$\Delta_S = \Delta_M$	0.14	0
w	1.2	0.6
c_l	2	0
c_g	0	2

Scenario 2 is presented to underline that the IPSO approach performs better than the DPSO approach. However, the graphs of the solutions for scenario 2 do not differ much from those of scenario 1, and consequently will be not reported.

All results are obtained considering a PC with a processor Intel® Core™ i7-2670QM CPU @ 2.20GHz and with 6.00 GB of RAM.

5.6.1 DPSO Results

The DPSO algorithm produces a solution with a sub-optimal final time; moreover, the boundary constraints are not completely satisfied. However, this kind of solution may be used as optimal initial guess to be given to a POCS, which may eventually find the optimal solution.

The only disadvantage is that the DPSO algorithm is strongly affected by the chosen geometry. The same algorithm may deliver excellent results with regards to

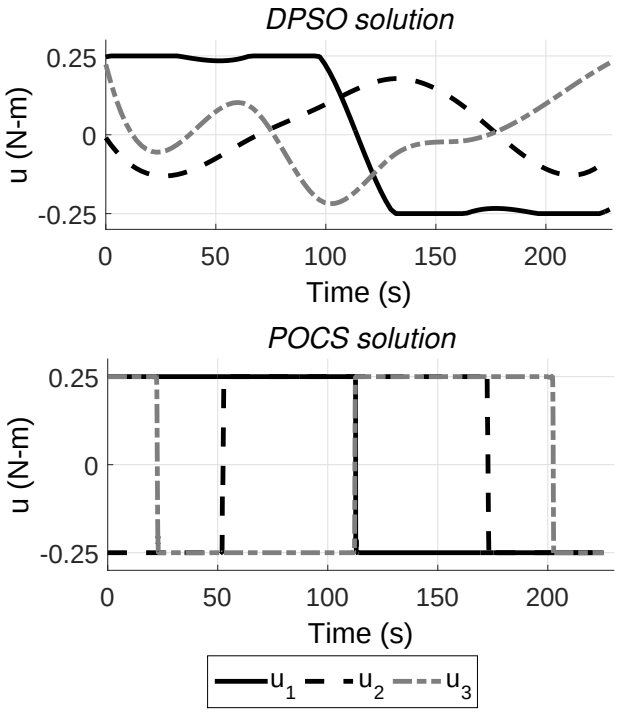


Fig. 5.4: Control policy from DPSO and POCS, scenario 1.

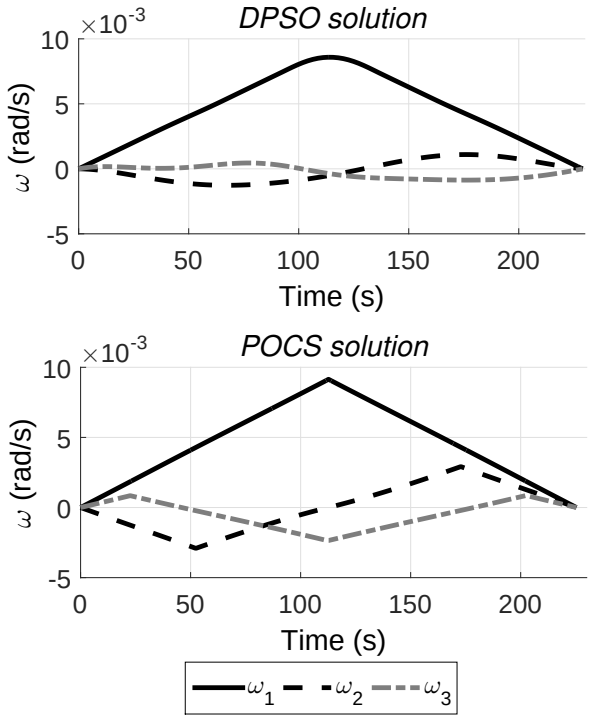


Fig. 5.5: Angular velocity history from DPSO and POCS, scenario 1.

the fulfillment of the boundary conditions for one geometry, and poor results for another one.

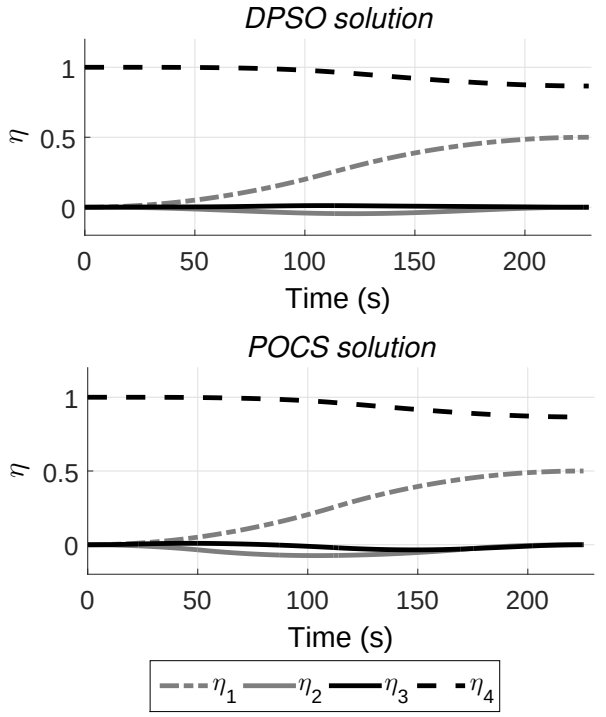


Fig. 5.6: Quaternions history from DPSO and POCS, scenario 1.

Tab. 5.5: Final state values for a sample experiment with DPSO, scenario 1.

	Units	Final value	Expected final value	Absolute error
ω_1	(rad/s)	$-2.5061 \cdot 10^{-6}$	0	$-2.5061 \cdot 10^{-6}$
ω_2	(rad/s)	$-8.3160 \cdot 10^{-7}$	0	$-8.3160 \cdot 10^{-7}$
ω_3	(rad/s)	$-1.9921 \cdot 10^{-7}$	0	$-1.9921 \cdot 10^{-7}$
η_1	(-)	0.4998	0.5	0.0002
η_2	(-)	$-2.8491 \cdot 10^{-4}$	0	$-2.8491 \cdot 10^{-4}$
η_3	(-)	$2.1350 \cdot 10^{-4}$	0	$2.1350 \cdot 10^{-4}$
η_4	(-)	0.8661	0.8660	0.0001
t_f	(s)	228.6434	225.3324	3.3110

The results obtained in case 1 are shown in Fig. 5.4, Fig. 5.5 and Fig. 5.6, where they are compared with the optimal solution given by the POCS. In this case, both the angular displacement and the angular velocity reach the desired final value with acceptable tolerance values. The errors on the boundary constraints are reported in Table 5.5 where the results of a generic simulation are chosen. From the comparison with the POCS optimal solution, it is clear that the result is in agreement with the optimal solution.

In Table 5.6, ten example results obtained with the DPSO are reported. T refers to the computational time, t_f is the obtained maneuver time and ϵ represents the percentage relative error with respect to the POCS optimal time $t_f^* = 225.33$ s. The

Tab. 5.6: List of 10 results obtained with DPSO, scenario 1.

Case	T_{DPSO} (s)	t_f (s)	ϵ (-)	$\Delta\theta$ (deg)	$\Delta\|\boldsymbol{\omega}\|$ (deg/s)
1	217.51	228.19	1.27 %	0.12	0.05
2	263.98	234.83	4.22 %	0.29	0.09
3	284.46	228.11	1.23 %	0.08	0.03
4	242.52	225.22	-0.05 %	2.10	0.87
5	218.06	249.17	10.58 %	0.06	0.02
6	190.74	240.22	6.61 %	0.75	0.27
7	216.35	225.10	-0.10 %	2.34	0.71
8	216.87	228.51	1.41 %	0.05	0.03
9	248.90	228.09	1.22 %	0.08	0.06
10	221.10	235.95	4.71 %	0.05	0.03

Tab. 5.7: Final state values from a sample experiment obtained with DPSO, scenario 2.

	Units	Final value	Expected final value	Absolute error
ω_1	(rad/s)	$1.7812 \cdot 10^{-4}$	0	$1.7812 \cdot 10^{-4}$
ω_2	(rad/s)	$1.7981 \cdot 10^{-4}$	0	$1.7981 \cdot 10^{-4}$
ω_3	(rad/s)	$-1.3373 \cdot 10^{-4}$	0	$1.3373 \cdot 10^{-4}$
η_1	(-)	0.4804	0.5	0.0196
η_2	(-)	-0.0200	0	0.0200
η_3	(-)	-0.0070	0	0.0070
η_4	(-)	0.8768	0.8660	0.0108
t_f	(s)	222.1578	225.9021	3.7443

value $\Delta\theta$ is the difference between the obtained angle of rotation θ and the imposed Θ_f while $\Delta\|\boldsymbol{\omega}\|$ is the norm of the error of the final angular velocity. With the DPSO approach, the maneuver time is affected by highly variable relative errors. Moreover, the tests *4* and *7* show that, when t_f is in proximity with t_f^*, the rotation angle is affected by a relatively high error. Furthermore, the required computational time varies from one test to another. This unstable trend of results makes the generic DPSO solution unreliable.

When case 2 is analyzed with DPSO, the results are poorer than those obtained in the first case. The errors on the boundary constraints are reported in Table 5.7: it can be seen that the errors are greater than those obtained in case 1. The maneuver time t_f, however, is as positive as in those solutions of the first case.

The solutions in cases 1 and 2 using the DPSO approach cannot be used as stand-alone solutions unless a further control system improves the final time of the maneuver. However, the DPSO solution may always be used as the initial guess for a POCS.

5.6.2 Basic IPSO results

In this section the results obtained with the IPSO are shown. Note that these results are related to the first version of the IPSO, and improved results are reported in the following sections.

The advantage of the IPSO approach may be evaluated comparing its solution with the DPSO solution. In Fig. 5.7, Fig. 5.9 and Fig. 5.8, where case 1 is studied, the IPSO solution is compared with the POCS solution. The IPSO result captures all the essential characteristics of the optimal maneuver. Although the IPSO algorithm uses the Modified Rodrigues Parameter, the angular displacement is described using the quaternions, in order to easily compare the solutions offered by the different methods. Therefore, for example, the IPSO control shown in Fig. 5.7 is positive on each axis when the optimal maneuver requires a positive bang, and negative when the optimal maneuver requires a negative bang. The trend of the quaternions (Fig. 5.9) and that of the angular velocity (Fig. 5.8) is still closer to the optimal one. Both the angular displacement (i.e. the values of the quaternion) and the angular velocity reach the imposed final value. In fact, the most important difference between the IPSO and the DPSO algorithm is that the IPSO satisfies completely all the boundary conditions since they are imposed at the beginning of the algorithm.

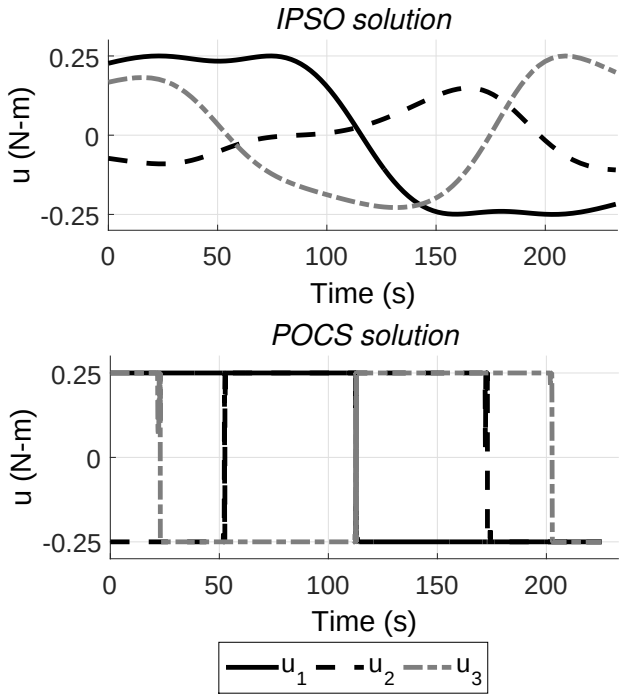


Fig. 5.7: Control policy from basic IPSO and POCS, scenario 1.

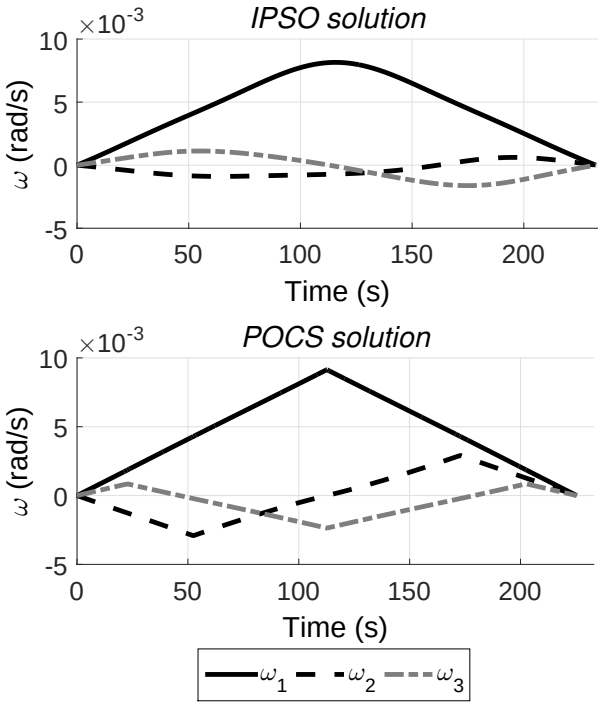


Fig. 5.8: Angular velocity history from basic IPSO and POCS, scenario 1.

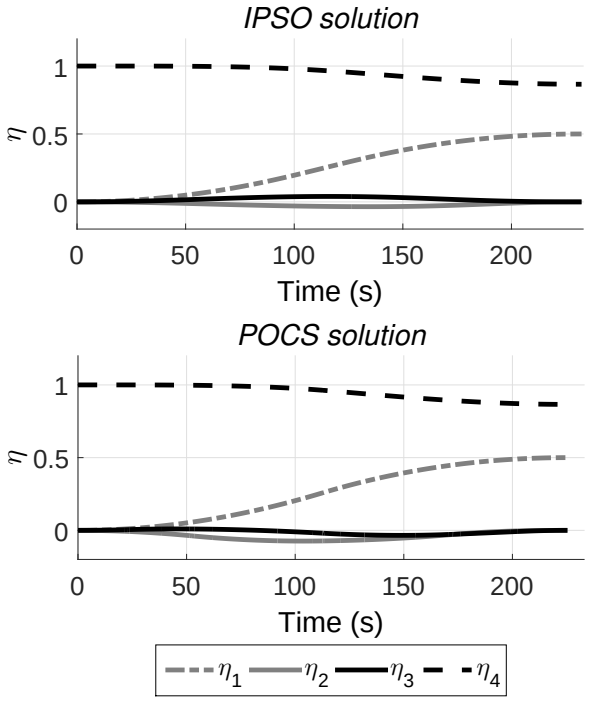


Fig. 5.9: Quaternions history from basic IPSO and POCS, scenario 1.

The evolution of the swarm is reported through Fig. 5.10, Fig. 5.11 and Fig. 5.12, where the maneuver is shown in the inertial reference frame in terms of latitude and longitude. The reference frame is rotated in order to make the figures easier to

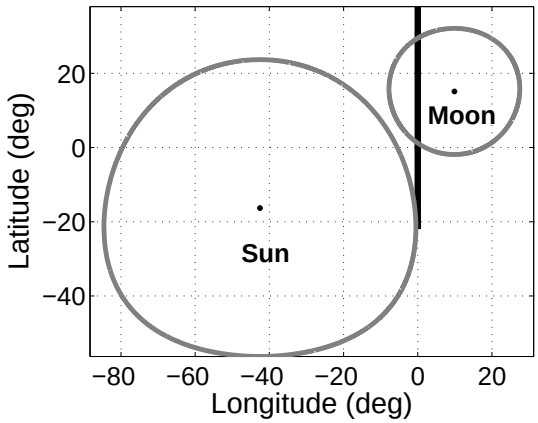


Fig. 5.10: Sensor path at the initialization of the basic IPSO algorithm, scenario 2.

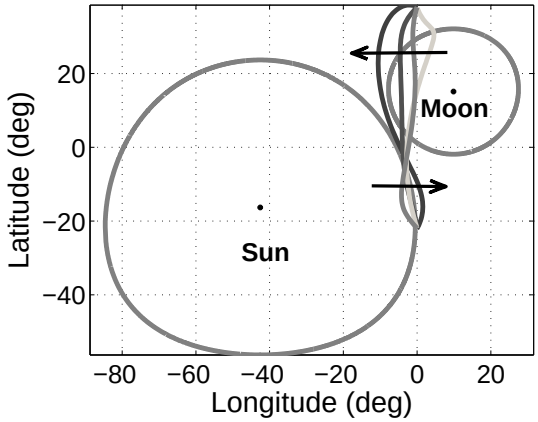


Fig. 5.11: Evolution of the sensor path associated to the global best particle, scenario 2.

read, i.e. the axes of the keep-out cones are brought near the equator axis (where the latitude is equal to zero) to minimize the deformation of the cones.

As it can be seen in Fig. 5.10, in the first step, the sensor axis goes from A to B without considering the two keep-out cones and passing through them. The idea used in the simulation is that one of the particles of the swarm is initialized with a linear interpolation between the initial and the final values of the Modified Rodriguez Parameters, while for the value of the final time it is not required to select particular value. This expedient guarantees that all particles enter the feasible search space in the first cycle of the evolution, when the keep-out cones are not contemplated.

In the following cycles (Fig. 5.11), the swarm moves in such a way that the keep-out cones constraint is progressively satisfied: the global best particle moves from A to B in order to minimize the maneuver time and the inclusion of the keep-out cones simultaneously. The arrows and the different colors of the maneuvers show

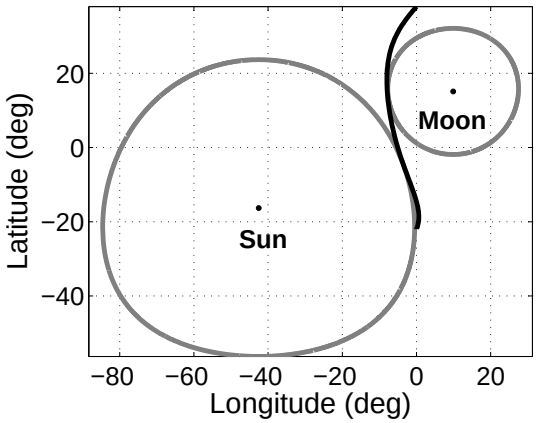


Fig. 5.12: Optimal sensor path at the end of the evolution, scenario 2.

the direction of the evolution. The progressive removal of the keep-out cones is implemented to make the evolution faster. In fact, only a little alteration of the global best is needed in order to completely satisfy the renewed tolerance value.

Finally, as it can be seen in Fig. 5.12, the swarm satisfies completely the keep-out cones constraint and evolves ito minimize the maneuver time. With regards to the kinematics, no relevant differences are to be seen between the IPSO and the POCS solutions. The most important criterion, which is to prevent the axis from entering the keep-out cones, is a common characteristic of all the geometries studied with the IPSO approach.

We observe an important feature of the IPSO through the different numerical simulations: even when the evaluation of the direction of the trajectory is incorrect in the first cycles (i.e. the maneuver does not pass between the cones), the swarm is able to change direction and set the new global best particle thanks to the division of the swarm in several search groups.

With regards to the computational time, a list of 10 different experiments carried out for case 1 are reported in Table 5.8. As before, T refers to the computational time, t_f refers to the maneuver time and ϵ represents the relative error with respect to the POCS optimal time $t_f^* = 225.33$ s. The following important remarks may be pointed out:

- The computational time does not substantially change from one experiment to another.

- The computational time has been halved with respect to the computational time required by the DPSO algorithm.

Tab. 5.8: List of 10 results obtained with IPSO, scenario 1.

Case	T_{IPSO} (s)	t_f (s)	ϵ
1	107.60	232.23	3.06 %
2	107.15	232.03	2.97 %
3	107.85	232.09	3.00 %
4	108.53	232.12	3.01 %
5	108.09	231.93	2.93 %
6	108.48	232.45	3.16 %
7	107.62	231.92	2.92 %
8	108.15	233.11	3.45 %
9	108.11	232.33	3.11 %
10	108.10	232.85	3.34 %

- The error between the obtained final time and the optimal one (225.33 s) is always about 3%, which is a very reasonable result. The unstable trend of the DPSO approach is removed, making the IPSO a more reliable solution.

With regards to the solution in case 2, IPSO provides the same results as in case 1. The computational time is equal to the computational time in case 1. The boundary constraints are completely satisfied, as well as the constraint defined by the keep-out cones and the error of the final time is always about 3%. The performances are the same reported in Table 5.8, with errors sharing the same order of magnitude about the maneuver time t_f.

The strength of the IPSO lies in the fact that a near-optimal solution can be found for all geometries without requiring different computational times. As a final remark, it is important to underline that the IPSO solution is able to fully satisfy all the boundary and path constraints offering a maneuver time slightly greater than the optimal one. If the constraint about the optimal minimum time is relaxed and a near-optimal solution consistent with the constraints is accepted, then the IPSO solution is favorable considering the computational time required. Moreover, while no differences have been reported when analyzing cases 1 and 2 applying the IPSO approach, using a POCS, case 2 is more difficult to solve than case 1, because it requires greater computational times to obtain a positive solution.

5.6.3 Basic IPSO solution used as best guess

The IPSO approach may be used as:

1. Initial guess for a POCS, as the DPSO approach.

Tab. 5.9: List of 10 results obtained with the Hybrid Method technique, scenario 1.

Case	T_{IPSO} (s)	$t_{f,IPSO}$ (s)	T_{POCS} (s)	$t_{f,POCS}$ (s)	T_{TOT} (s)
1	107.60	232.23	178.17	225.33	285.77
2	107.15	232.03	61.23	225.33	168.38
3	107.85	232.09	203.68	225.33	311.53
4	108.53	232.12	67.78	225.33	176.31
5	108.09	231.93	87.86	225.33	195.95
6	108.48	232.45	75.76	225.33	184.24
7	107.62	231.92	107.78	225.33	215.40
8	108.15	233.11	88.60	225.33	196.74
9	108.11	232.33	93.31	225.33	201.42
10	108.10	232.85	207.39	225.33	315.49

2. Planner for near minimum-time maneuvers: in fact the algorithm guarantee that all constraints are satisfied. Moreover, the numerical results will prove that the maneuver time is very close to that obtained with the POCS approach.

If the computation of the exact optimal solution is required, the IPSO algorithm alone is not sufficient, since it only gives a near-optimal solution. As mentioned in the previous sections, the exact solution may be obtained using a POCS which often requires high computational times. Differently from the proposed PSO algorithms, which require always the same computational time regardless of the geometry, the POCS may significantly change in behavior depending on the studied geometry. In order to prevent such a problem, a hybrid method is proposed (such a strategy has already been proposed in [102]): a sub-optimal solution produced by IPSO is given as *best guess* to the POCS, which may now obtain the optimal solution in very low computational times. In this case, in fact, the POCS only needs to improve the IPSO solution, that is closer to the optimum. Furthermore, the DPSO may be used as the initial guess obtaining the same improvement in the computational time required by the POCS. However, the IPSO minimizes the total computational time, requiring less time to evaluate the sub-optimal solution than the DPSO. When no initial guess is given, the POCS creates the initial solution as a linear interpolation between the initial and the final values both for the state and the control.

It is important to underline that the behavior of the pseudospectral optimization software is strongly related to the required tolerance at the mesh points. When a bang-bang solution is the optimal solution, a high value of the tolerance is needed, and the computational time is quite high. For the reported experiments, the mesh tolerance is set to 10^{-11}. Obviously, when higher tolerances are required, the hybrid method is even more useful.

Tab. 5.10: List of 10 results obtained with the Hybrid Method technique, scenario 3.

Case	T_{IPSO} (s)	$t_{f,IPSO}$ (s)	T_{POCS} (s)	$t_{f,POCS}$ (s)	T_{TOT} (s)
1	99.32	443.82	150.06	404.71	249.38
2	100.42	452.61	141.63	404.71	242.05
3	99.61	447.78	175.41	404.71	275.01
4	99.79	449.83	346.00	404.71	445.79
5	96.50	443.39	252.11	404.71	348.61
6	96.49	453.79	234.74	404.71	331.22
7	100.22	443.63	259.96	404.71	360.18
8	105.58	446.79	188.97	404.71	294.56
9	99.75	456.66	151.34	404.71	251.09
10	100.20	453.49	197.74	404.71	297.94

With regards to scenario 1, the optimal solution is shown in Fig. 5.7, Fig. 5.9 and Fig. 5.8, where the behaviour of angular displacements, angular velocities and control are reported. As far as the computational time is concerned, the Table 5.9 reports the computational time required by the hybrid method. The total required time is, on average, 225.12 seconds; when using the POCS without the best guess, the computational time is on average 351.25 seconds. As it can be seen, the hybrid method leads to a timesavings of 35.91%. The advantage may be also seen in the mesh refinements and the total variables needed in order to obtain the optimal solution: their value is always higher when a best guess is not given.

For scenario 3 in Table 5.10, the computational time required by the hybrid method is on average 323.77 seconds, as reported in Table 5.10. When no best guess is used, the average computational time required by the POCS is 1522.57 seconds, which is almost five times the computational effort of the hybrid approach. This test has been carried out using the same satellite but with different positions of the sensor axis and different exclusion cones.

It may be important to underline that if, on one hand, the pseudospectral algorithm provides the optimal solution, on the other hand, it may require greater computational times. Sometimes, as for scenario 2, the optimal solution may be obtained with only very high tolerances.

5.7 Improved IPSO results

In this section the results from Ref. [2] are reported. In this section the definitive IPSO version is implemented, i.e. the one described in Sec. 4.1. When IPSO with improved B-spline approximation is employed, the particles are arrays containing:

- The value of the maneuver time t_f.

- The angular displacement parameters $\tilde{p}_{j,k}$ with $j = 1, 2, 3$ and $k = 1, 2, ..., N_{\mathcal{P}}$. As before, n is the number of scalars required by the chosen angular representation and $N_{\mathcal{P}}$ is the number of points used for the interpolation of the angular displacement.

- the time instants $\tilde{t}_{j,k}$. The kinematics is obtained in $N_T + 1$ points (so that $\tilde{\boldsymbol{t}} = [\tilde{t}_{i,1} = 0, \tilde{t}_{i,2} > \tilde{t}_{i,1}, ..., \tilde{t}_{i,N_{\mathcal{P}}-1} = 1]$) interpolating the $N_{\mathcal{P}}$ points $\mathbf{U}_{j,k} = [t_{j,k}, \tilde{p}_{j,k}]$ with B-splines.

The difference between the basic B-spline approximation is that the parameters $t_k^{(i)}$ are no more equally spaced but are variable used to shape the time B-spline mesh, as described in Sec. 4.4. Accordingly, the IPSO particle is defined as:

$$\mathbf{x} = [\tilde{\boldsymbol{p}}_1, ..., \tilde{\boldsymbol{p}}_{N_u}, \tilde{\boldsymbol{t}}_1, ..., \tilde{\boldsymbol{t}}_{N_u}, t_f] \in \mathbb{R}^{2N_u N_{\mathcal{P}}+1}. \tag{5.47}$$

In the following, it is shown that the improved B-spline approximation leads to more accurate solutions with respect to the basic B-spline approximation. This is due to the fact that each flat output parameter can vary its time mesh in order to improve the approximation of the optimal flat output trajectory.

Five different case studies are proposed, whose characteristics are reported in Table 5.11. The geometries of the first four scenarios are reported in Fig. 5.13. The directions of the Moon and the Sun are expressed in $\mathcal{B}_0$. The first and the second cases are different roll rotations with the minimum-time maneuver between the keep-out cones. The third case is a pitch rotation and the fourth case is a roll rotation where the minimum-time maneuver is not between the two cones. The maneuvers reported in Fig. 5.13 have been obtained with the IPSO approach. Note that case 1 is the same as case 1 in Sec. 5.6. Scenario 5 will be considered in Sec. 5.7.3 since it is a special case with zero free angle.

The exit criterion is the one described in Sec. 4.5.3 with $M = 10$, i.e.

$$\delta\overline{J}_g = \frac{1}{10} \sum_{i=\hat{k}-9}^{\hat{k}} \frac{\overline{J}_g^{(i-1)} - \overline{J}_g^{(i)}}{\overline{J}_g^{(i)}} < \varepsilon_c, \tag{5.48}$$

where the superscript N has not been reported and $\hat{k}$ is the iteration index where the global best particle is updated.

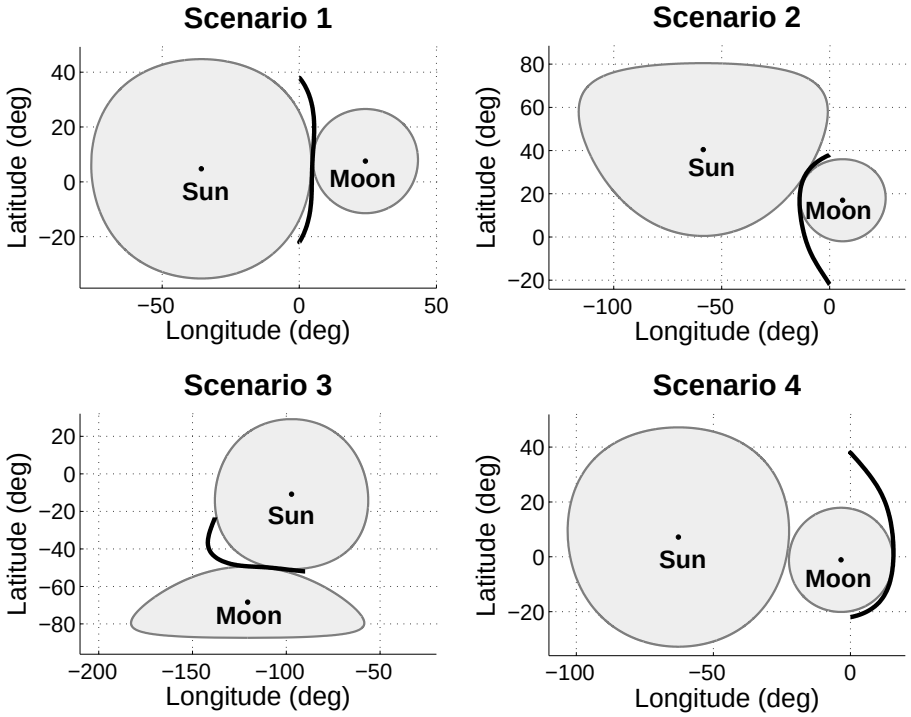


Fig. 5.13: Proposed scenarios for the improved IPSO tests.

5.7.1 Keep-out cone constraint

First of all it is interesting to report how the improved strategy for the decreasing tolerances (see Sec. 4.5.2 for details) affects the keep-out cones constraint whose evolution during the IPSO process is reported in Fig. 5.14. In the first step (subfigure 5.14.a), the keep-out cone constraint is not taken into account. As can be seen, one particle of the swarm may be initialized with a sensor trajectory going from the initial position to the final position along a straight line. In particular, the angular distance between the Sun and the Moon axes and this straight trajectory

Tab. 5.11: Direction of Sun and Moon in $\mathcal{B}_0$ for the proposed case studies with improved IPSO.

Scenario			Sun (in $\mathcal{B}_0$)			Moon (in $\mathcal{B}_0$)			Free Angle
N.	Θ_f (deg)	type	$\boldsymbol{\sigma}_{S,x}$	$\boldsymbol{\sigma}_{S,y}$	$\boldsymbol{\sigma}_{S,z}$	$\boldsymbol{\sigma}_{M,x}$	$\boldsymbol{\sigma}_{M,y}$	$\boldsymbol{\sigma}_{M,z}$	(deg)
1	60	roll	-0.58	-0.08	-0.81	0.41	-0.13	-0.91	0.52
2	60	roll	-0.65	-0.65	-0.40	0.10	-0.29	-0.95	0.84
3	60	pitch	-0.12	-0.98	-0.19	-0.19	-0.32	-0.93	0.60
4	60	roll	-0.88	-0.13	-0.45	-0.06	0.02	-0.99	0.78
5	60	roll	-0.58	-0.08	-0.81	0.29	-0.09	-0.95	0

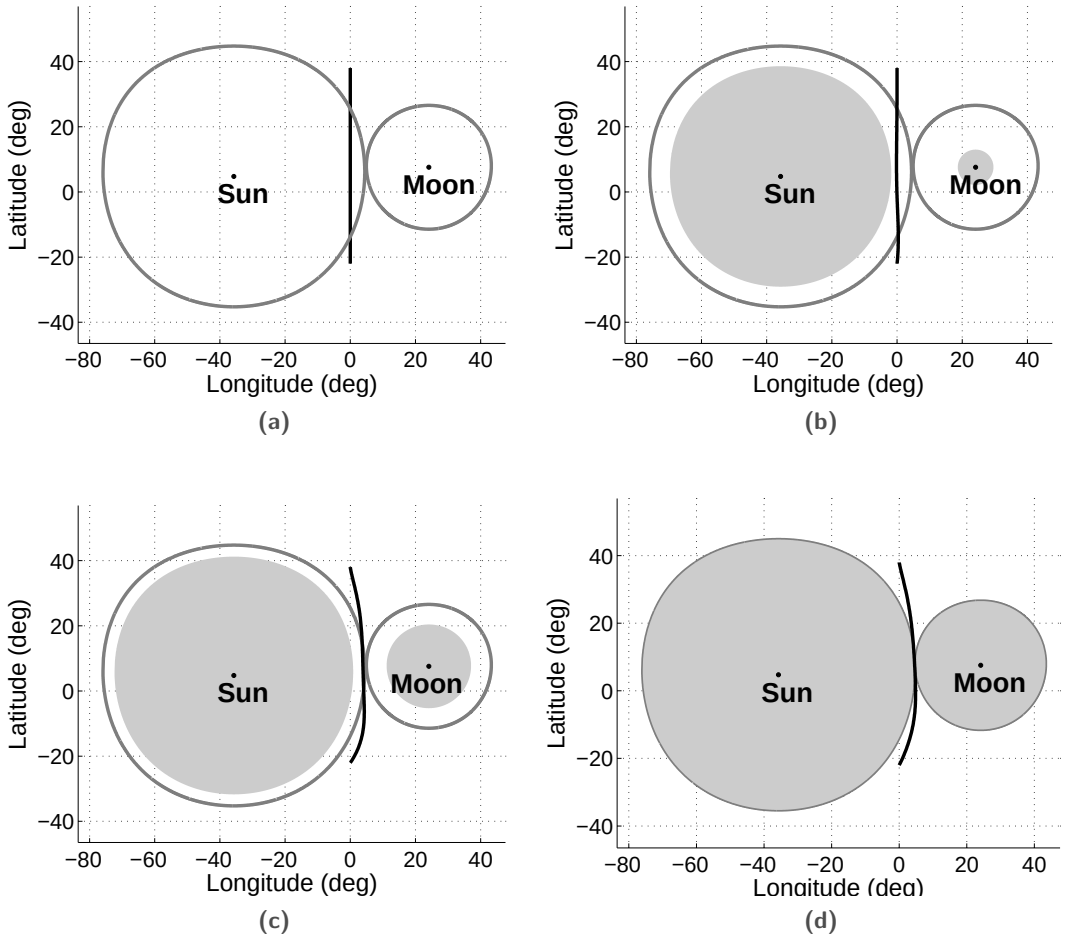


Fig. 5.14: Decreasing tolerances strategy for the keep-out cones, scenario 1.

may be evaluated and designated β_S^0 and β_M^0 according to Fig. 5.1. Considering the Sun keep-out constraint, the initial value of the keep-out tolerance may be set as

$$\Delta_S^{(0)} = \begin{cases} \epsilon_1 & \text{if } \beta_S^0 \geq \alpha_S \\ 1 - \cos\left(\alpha_S - \epsilon_2 \beta_S^0\right) & \text{if } \alpha_S > \beta_S^0 \geq \epsilon_3 \\ 1 - \cos\,\alpha_S & \text{if } \beta_S^0 < \epsilon_3 \end{cases} \tag{5.49}$$

where, for this work, $\epsilon_1 = 0.05$, $\epsilon_2 = 0.8$ and $\epsilon_3 = 10$ degree. The initial value of the Moon keep-out tolerance is evaluated in the same way.

As can be seen from Fig. 5.14, the keep-out constraints are are slowly introduced in the optimization process (sub-figures 5.14.b and 5.14.c) until they are completely satisfied as in 5.14.d. Though other initial values of the keep-out constraint tolerances may be chosen, it is of the utmost importance to underline that the decreasing tolerances strategy leads to a very straightforward evolution of the swarm which

is difficult to achieve when the cone constraints are completely taken into account from the beginning of the evolution.

5.7.2 Free angle between keep-out cones

All the results obtained with the proposed approach have been compared with the results obtained with a POCS. In Fig. 5.15 the percentage error of the IPSO otimal time with respect to the POCS optimal time is reported after having carried out 600 experiments. As it can be seen, case 1 and case 3 have a mean error of about 2%, while in the other two cases we arrive at a maximum mean error of about 6%. It can be seen that, for case 1, the performances have been improved with respect to those obtained with the basic B-spline approximation in Sec. 5.6.

The most important characteristic of the proposed approach is that the solution is always around the POCS solution, i.e. the problem of local minima associated with other possible trajectory around the exclusion cone is completely avoided. This is particularly important for cases 2 and 4, where a local minimum with final time close to the obtained minimum time exists on the opposite side of the reported maneuver (Fig. 5.13).

Detailed results have been reported for scenario 1 choosing one reference experiment. From Fig. 5.16, 5.17 and 5.18 we can see that the IPSO solution along the x axis is quite identical to the POCS solution. In this case, the maneuver is mainly

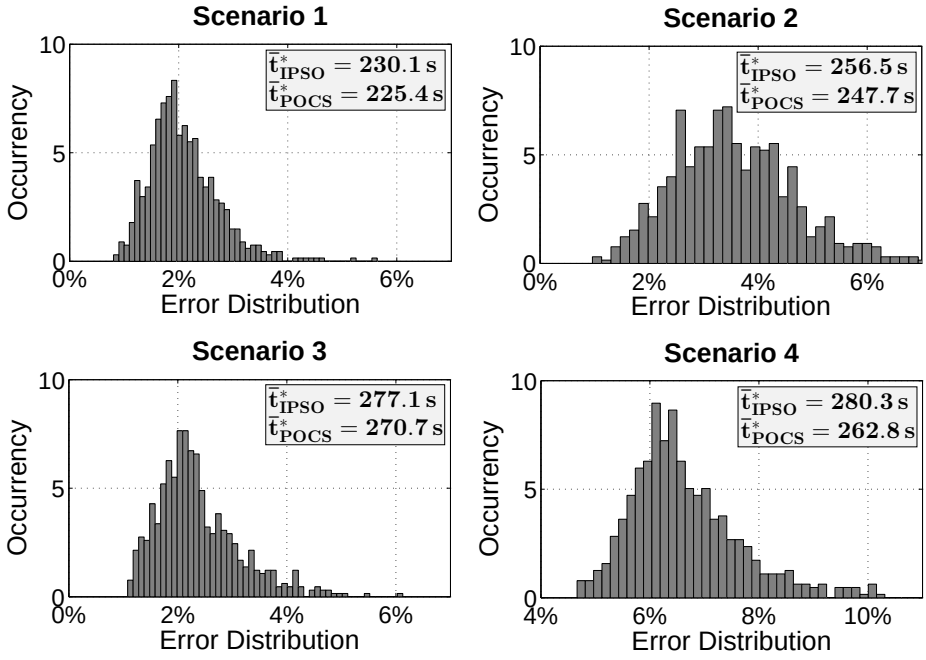


Fig. 5.15: Distribution of the improved IPSO results (600 test cases).

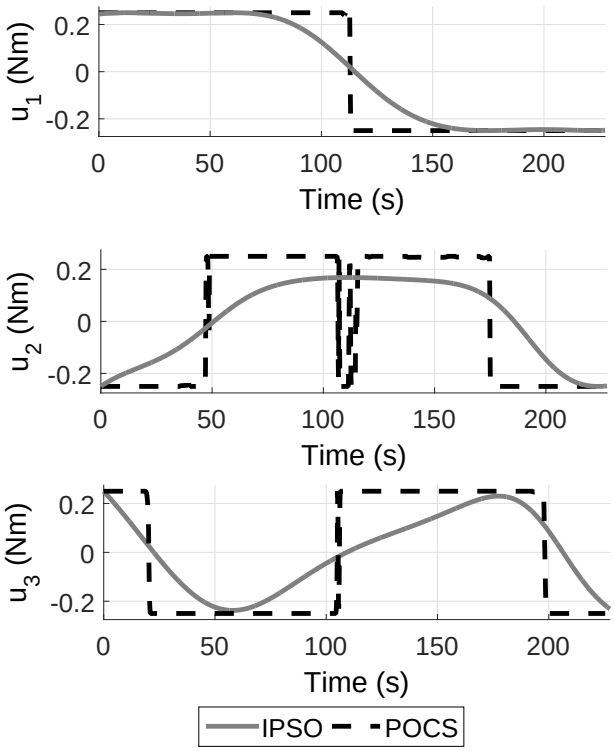


Fig. 5.16: Control policy from improved IPSO and POCS, scenario 1.

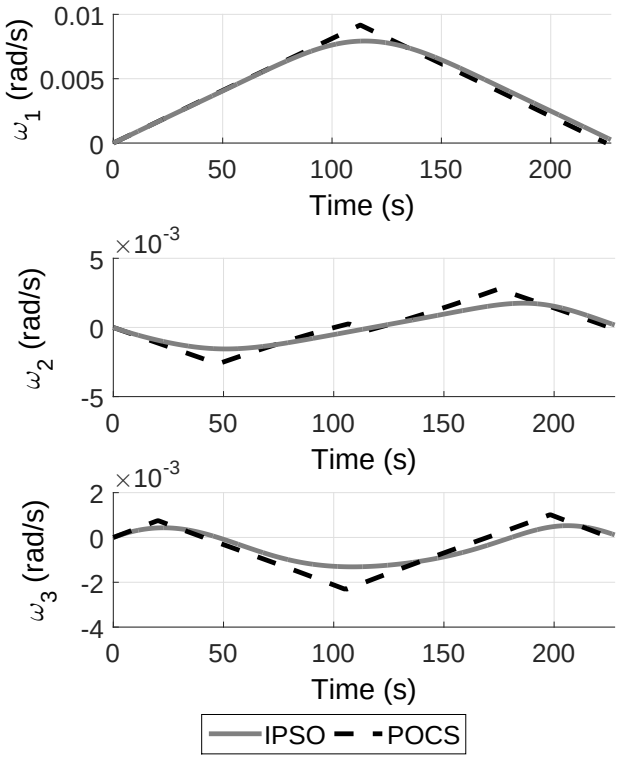


Fig. 5.17: Angular velocity history from improved IPSO and POCS, scenario 1.

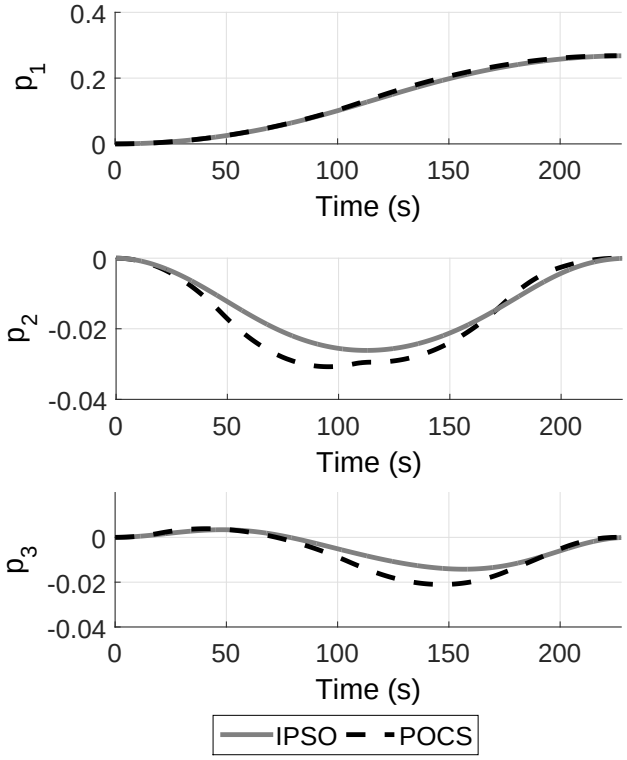


Fig. 5.18: MRPs history from improved IPSO and POCS, scenario 1.

along the $\boldsymbol{x}_{\mathcal{B}}$ axis: this result means that the main characteristics of the maneuver have been caught from the IPSO solution. The $\boldsymbol{y}_{\mathcal{B}}$ and $\boldsymbol{z}_{\mathcal{B}}$ axes show IPSO trends that differ from the POCS ones. It must be noted that all the constraints are satisfied by both the IPSO and POCS solutions.

With a refinement of the implemented code, the mean computational times required by the proposed IPSO approach has been reduced with respect to the one reported in Sec. 5.6: for the reported test cases about 50 seconds are required for the obtainment of the solution. This time does not depend on the particular geometry of the analyzed cases. A further reduction of the computational time will be one of the goal of the future development of the algorithm.

Accordingly with previous works in literature (e.g, see Ref. [102]) and similarly to Sec. 5.6.3, it has been noted that, using the IPSO solution as best guess for the POCS, computational times may be considerably reduced. For example, solving case 2 with the IPSO guess requires about 70 seconds, while about 3800 seconds are required without best guess.

5.7.3 No free angle between keep-out cones

In the previous section the reliability of the IPSO has been demonstrated through 4 different scenarios. In this section, scenario 5 in Table 5.11 is investigated to show the maneuver that can be evaluated when the two keep-out cones intersect each other. In this case, the parameter f introduced in Eq. (5.63) plays a fundamental role in understanding the feasibility of the PSO particle. We remind the reader that the method requires a complete knowledge of the keep-out cones geometries, so that the parameter f may be evaluated and used to compute the performance index of the particles.

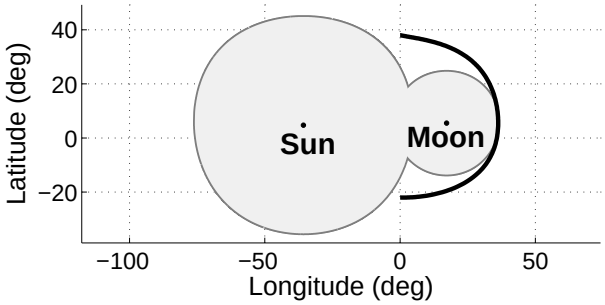


Fig. 5.19: IPSO trajectory with no free angle, scenario 5.

The geometry considered in this section is scenario 5 reported in Table 5.11. The reader can verify that no free angle is left between the cones. The near-optimal IPSO solution is reported in Fig. 5.19. The maneuver takes 430.75 s. The IPSO performances are consistent with the ones reported in the previous section. Also in this case, moreover, the control resembles a bang-bang policy.

5.8 Spacecraft modelled with reaction wheels

This chapter examines a new numerical solution based on IPSO to determine the approximate solutions for a constrained, time-optimal satellite reorientation problem accomplished with reaction wheels. The theory and the results reported in the following are taken from the author's paper, Ref. [3]. Reaction wheels are commonly used for satellite attitude control and they can be mathematically modeled as internal torques in Euler's equation of rigid-body motion.

So far, several works considering slew maneuvers with reaction wheels may be found in the literature, starting from papers dating back to the 1990's [115, 116] and continuing to recent years [117, 118]. The introduction of the wheel dynamics requires taking into account the conservation of the total inertial angular momentum and the saturation of the wheel velocity and acceleration.

In this section, the problem described in Sec. 5.2 is modeled with more details introducing the internal torques of the wheels. The inverse dynamics approach is exploited and, accordingly, the object of IPSO evolution is the kinematics rather than the control. As a result, the final boundary constraints are exactly satisfied. However, the proper modeling of the wheels requires the integration of only a part of the equation of motion: in this chapter this issue has been tackled involving a fixed-step integrator.

In this chapter the definitive version of the IPSO has been used, i.e. the one described in Chapter 4. Accordingly, the improved B-spline approximation is employed and adaptive decreasing tolerances are used.

5.8.1 Dynamical model

In a satellite-fixed reference frame $\mathcal{S}$ placed at the CM, we define the satellite angular momentum as $\boldsymbol{H}^{\mathcal{S}} = I^{\mathcal{S}}\boldsymbol{\omega}^{\mathcal{S}/\mathcal{I}}$, where $\boldsymbol{\omega}^{\mathcal{S}/\mathcal{I}}$ is the angular velocity of the satellite with respect to the inertial frame $\mathcal{I}$ expressed in $\mathcal{S}$. If reaction wheels are used to accomplish the maneuvers, we can define the total angular momentum $\boldsymbol{H}^{t} = \boldsymbol{H}^{\mathcal{S}} + \boldsymbol{H}^{\mathcal{W}}$, with $\boldsymbol{H}^{\mathcal{W}} = I^{\mathcal{W}}\boldsymbol{\omega}^{\mathcal{W}/\mathcal{I}}$ (the angular momentum of the wheels) where $\boldsymbol{\omega}$ì$^{\mathcal{W}/\mathcal{I}}$ is the angular velocity of the wheels with respect to $\mathcal{I}$ expressed in $\mathcal{S}$. The free rigid-body motion of a satellite equipped with reaction wheels is described by Euler's equation [119]

$$\dot{\boldsymbol{H}}^{t} + \boldsymbol{\omega}^{\mathcal{S}/\mathcal{I}} \times \boldsymbol{H}^{t} = \boldsymbol{0} \tag{5.50}$$

that can be split into the satellite and wheel contributions:

$$\dot{\boldsymbol{H}}^{\mathcal{S}} + \boldsymbol{\omega}^{\mathcal{S}/\mathcal{I}} \times \boldsymbol{H}^{\mathcal{S}} = -\dot{\boldsymbol{H}}^{\mathcal{W}} - \boldsymbol{\omega}^{\mathcal{S}/\mathcal{I}} \times \boldsymbol{H}^{\mathcal{W}}. \tag{5.51}$$

The rhs of Eq. (5.51) properly defines the internal torques $\boldsymbol{T}_{int}$ of the system, i.e.

$$\dot{\boldsymbol{H}}^{\mathcal{W}} + \boldsymbol{\omega}^{\mathcal{S}/\mathcal{I}} \times \boldsymbol{H}^{\mathcal{W}} = -\boldsymbol{T}_{int}\,. \tag{5.52}$$

The motion of the wheels influences the motion of the satellite by 1) changing the speed of the wheels through an electric motor (1^{st} lhs term in Eq. (5.52)) and 2) changing the orientation of the wheels with respect to $\mathcal{I}$ (2^{nd} lhs term in Eq. (5.52)). Note that the second term is a gyroscopic term due to the motion of the satellite.

When a body reference system $\boldsymbol{\mathcal{B}}$ aligned with the principal inertia axes is chosen in $\mathcal{S}$ such as $\boldsymbol{\mathcal{B}} = \{\hat{\boldsymbol{e}}_x, \hat{\boldsymbol{e}}_y, \hat{\boldsymbol{e}}_z\}$, the inertia tensor $I^{\mathcal{S}}$ is diagonal and the satellite angular momentum $\boldsymbol{H}^{\mathcal{S}}$ is

$$\boldsymbol{H}^{\mathcal{S}} = [I_x^{\mathcal{S}}\omega_x^{\mathcal{S}/\mathcal{I}} \;\; I_y^{\mathcal{S}}\omega_y^{\mathcal{S}/\mathcal{I}} \;\; I_z^{\mathcal{S}}\omega_z^{\mathcal{S}/\mathcal{I}}]^T\,. \tag{5.53}$$

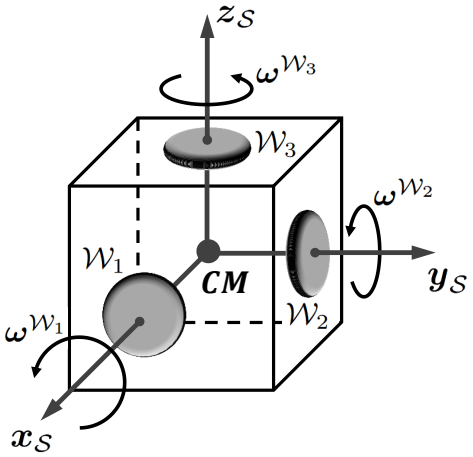


Fig. 5.20: Spacecraft model with reaction wheels.

Moreover, let $= \{\boldsymbol{x}_{\mathcal{S}}, \boldsymbol{y}_{\mathcal{S}}, \boldsymbol{z}_{\mathcal{S}}\}$ be the orthonormal axes defined in $\boldsymbol{\mathcal{B}}$. Finally, $\boldsymbol{\mathcal{B}}_0 = \{\hat{\boldsymbol{e}}_{0,x}, \hat{\boldsymbol{e}}_{0,y}, \hat{\boldsymbol{e}}_{0,z}\}$ is an inertial reference system coincident with $\boldsymbol{\mathcal{B}}$ at $t = t_0$.

Without loss of generality, let us assume that three reaction wheels labeled as $\mathcal{W}_1$, $\mathcal{W}_2$ and $\mathcal{W}_3$ with the same polar moment of inertia $I^{\mathcal{W}}$ are exploited and that they are aligned with the reference system axes as reported in Fig. 5.20. Accordingly, denoting with $\omega^{\mathcal{W}_1/\mathcal{I}}$, $\omega^{\mathcal{W}_2/\mathcal{I}}$ and $\omega^{\mathcal{W}_3/\mathcal{I}}$ the norm of the angular velocities of the three wheels, we can define the total angular velocity of the wheels as

$$\boldsymbol{\omega}^{\mathcal{W}/\mathcal{I}} = \begin{bmatrix} \omega^{\mathcal{W}_1/\mathcal{I}} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \omega^{\mathcal{W}_2/\mathcal{I}} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega^{\mathcal{W}_3/\mathcal{I}} \end{bmatrix} = \begin{bmatrix} \omega_x^{\mathcal{W}/\mathcal{I}} \\ \omega_y^{\mathcal{W}/\mathcal{I}} \\ \omega_z^{\mathcal{W}/\mathcal{I}} \end{bmatrix} \tag{5.54}$$

and $\boldsymbol{H}^{\mathcal{W}} = I^{\mathcal{W}}\boldsymbol{\omega}^{\mathcal{W}/\mathcal{I}} = I^{\mathcal{W}}\left(\boldsymbol{\omega}^{\mathcal{W}/\mathcal{S}} + \boldsymbol{\omega}^{\mathcal{S}/\mathcal{I}}\right)$ takes the following form:

$$\boldsymbol{H}^{\mathcal{W}} = \begin{bmatrix} I^{\mathcal{W}} & 0 & 0 \\ 0 & I^{\mathcal{W}} & 0 \\ 0 & 0 & I^{\mathcal{W}} \end{bmatrix} \begin{bmatrix} \omega_x^{\mathcal{W}/\mathcal{S}} + \omega_x^{\mathcal{S}/\mathcal{I}} \\ \omega_y^{\mathcal{W}/\mathcal{S}} + \omega_y^{\mathcal{S}/\mathcal{I}} \\ \omega_z^{\mathcal{W}/\mathcal{S}} + \omega_z^{\mathcal{S}/\mathcal{I}} \end{bmatrix}. \tag{5.55}$$

Simplifying the notation and using $\mathcal{W}$ and $\mathcal{S}$ to denote $\mathcal{W}/\mathcal{S}$ and $\mathcal{S}/\mathcal{I}$ and introducing Eq. (5.53) and Eq. (5.55) into Eq. (5.51) yields

$$\begin{aligned} I^{\mathcal{S}}\dot{\boldsymbol{\omega}}^{\mathcal{S}} + \boldsymbol{\omega}^{\mathcal{S}} \times I^{\mathcal{S}}\boldsymbol{\omega}^{\mathcal{S}} = \\ -I^{\mathcal{W}}\dot{\boldsymbol{\omega}}^{\mathcal{W}} - I^{\mathcal{W}}\dot{\boldsymbol{\omega}}^{\mathcal{S}} - \boldsymbol{\omega}^{\mathcal{S}} \times I^{\mathcal{W}}\boldsymbol{\omega}^{\mathcal{W}} - \boldsymbol{\omega}^{\mathcal{S}} \times I^{\mathcal{W}}\boldsymbol{\omega}^{\mathcal{S}}. \end{aligned} \tag{5.56}$$

Rearranging Eq. (5.56) and defining $I^t = I^{\mathcal{S}} + I^{\mathcal{W}}$ we obtain

$$I^t \dot{\boldsymbol{\omega}}^{\mathcal{S}} + \boldsymbol{\omega}^{\mathcal{S}} \times I^t \boldsymbol{\omega}^{\mathcal{S}} = -I^{\mathcal{W}} \dot{\boldsymbol{\omega}}^{\mathcal{W}} - \boldsymbol{\omega}^{\mathcal{S}} \times I^{\mathcal{W}} \boldsymbol{\omega}^{\mathcal{W}} \,. \tag{5.57}$$

Finally, Eq. (5.57) can be split as

$$I^t \dot{\boldsymbol{\omega}}^{\mathcal{S}} + \boldsymbol{\omega}^{\mathcal{S}} \times I^t \boldsymbol{\omega}^{\mathcal{S}} = \tilde{\boldsymbol{T}}_{int} \,, \tag{5.58a}$$

$$I^{\mathcal{W}} \dot{\boldsymbol{\omega}}^{\mathcal{W}} + \boldsymbol{\omega}^{\mathcal{S}} \times I^{\mathcal{W}} \boldsymbol{\omega}^{\mathcal{W}} = -\tilde{\boldsymbol{T}}_{int} \,, \tag{5.58b}$$

where the difference of $\tilde{\boldsymbol{T}}_{int}$ from $\boldsymbol{T}_{int}$ comes from a comparison of Eq. (5.58a) with Eq.(5.52).

5.8.2 Minimum-time slew maneuver problem

The problem is formulated as a Mayer optimal control problem [25], with performance index $J = t_f - t_0$, where $t_0 = 0$. The optimization problem may be summarized as follows:

$$\begin{aligned}
&\text{Find } \boldsymbol{p}(t), \boldsymbol{\omega}^{\mathcal{S}}(t), \boldsymbol{\omega}^{\mathcal{W}}(t), t_f \in \mathbb{R} \\
&\text{minimizing} \\
&J = t_f - t_0 \\
&\text{subject to, } \forall t \in [t_0, t_f] \\
&\text{dynamic constraints:} \quad I^t \dot{\boldsymbol{\omega}}^{\mathcal{S}} + \boldsymbol{\omega}^{\mathcal{S}} \times I^t \boldsymbol{\omega}^{\mathcal{S}} = \tilde{\boldsymbol{T}}_{int} \\
&\qquad\qquad\qquad\quad I^{\mathcal{W}} \dot{\boldsymbol{\omega}}^{\mathcal{W}} + \boldsymbol{\omega}^{\mathcal{S}} \times I^{\mathcal{W}} \boldsymbol{\omega}^{\mathcal{W}} = -\tilde{\boldsymbol{T}}_{int} \\
&\text{kinematic constraints:} \quad \boldsymbol{\omega}^{\mathcal{S}} = 4\Psi^{-1}(\boldsymbol{p})\dot{\boldsymbol{p}} \\
&\qquad\qquad\qquad\quad \dot{\boldsymbol{\omega}}^{\mathcal{S}} = 4(\dot{\Psi}^{-1}(\boldsymbol{p})\,\dot{\boldsymbol{p}} + \Psi^{-1}(\boldsymbol{p})\,\ddot{\boldsymbol{p}}) \\
&\text{initial conditions:} \quad \mathbf{p}(t_0) - \mathbf{p}_0 = \mathbf{0}\,,\ \dot{\mathbf{p}}(t_0) = \mathbf{0} \\
&\qquad\qquad\qquad\quad \boldsymbol{\omega}^{\mathcal{W}}(t_0) = \mathbf{0} \\
&\text{final conditions:} \quad \mathbf{p}(t_f) - \mathbf{p}_f = \mathbf{0}\,,\ \dot{\mathbf{p}}(t_f) = \mathbf{0} \\
&\text{state path constraint:} \quad \boldsymbol{\sigma}(t) \cdot \boldsymbol{\sigma}_s - \cos(\alpha_s) \leq 0 \\
&\text{control path constraints:} \quad \left\| \dot{\boldsymbol{H}}^{\mathcal{W}}(t) \right\|_\infty \leq \dot{H}^{\mathcal{W}}_{max} \\
&\qquad\qquad\qquad\quad \left\| \boldsymbol{\omega}^{\mathcal{W}/\mathcal{S}}(t) \right\|_\infty \leq \omega^{\mathcal{W}}_{max}
\end{aligned} \tag{5.59}$$

The dynamic constraints have been described in Sec. 5.8.1, whereas the kinematic constraints are the same as in the previous sections. Initial and final conditions are imposed in order to deal with a *rest-to-rest* maneuver, and the path constraint is the keep-out-cone constraint introduced in Sec. 5.2.

For the proposed problem, the control constraints are imposed as saturation constraints on the torques and the angular momentum of the reaction wheels. The torque constraint is linked to the term $\dot{\boldsymbol{H}}^{\mathcal{W}}$ and not to $\boldsymbol{T}_{int}$, consistent with what was discussed after Eq. (5.52). The angular momentum constraint is applied to $\boldsymbol{\omega}^{\mathcal{W}/\mathcal{S}}$, i.e. to the wheels' angular velocities with respect to to the satellite reference frame.

5.8.3 IPSO transcription

The IPSO transcription of the problem is similar to the one proposed in Sec. 5.7. Indeed, the improved IPSO method is applied and the PSO particle is defined as

$$\mathbf{x} = [\tilde{\boldsymbol{p}}_1, ..., \tilde{\boldsymbol{p}}_{N_u}, \tilde{\boldsymbol{t}}_1, ..., \tilde{\boldsymbol{t}}_{N_u}, t_f] \in \mathbb{R}^{2N_u N_{\mathcal{P}}+1}. \tag{5.60}$$

As can be seen, the IPSO is applied only to the satellite kinematics. Actually, the approach that is described here represents an *hybrid* application of the IPSO, since there is a simultaneous application of inverse and direct approaches. The satellite dynamics is solved via inverse method, whereas the wheels' dynamics is solved via direct method, i.e. with numerical integration.

As a consequence, the transcribed optimal control problem is reported in Eq. (5.61).

$$\begin{aligned}
&\text{Find } \boldsymbol{p}^N(t) : t \to \mathbb{Y} = \mathbb{R}^{N_u} \text{ such that } \boldsymbol{p}^N \in \mathscr{Y}_B,\ t_f \in \mathbb{R} \\
&\text{minimizing} \\
&J = t_f - t_0 \\
&\text{subject to, } \forall t \in [t_0, t_f] \\
&\text{dynamic constraints:} \quad I^{\mathcal{W}}\dot{\boldsymbol{\omega}}^{\mathcal{W}} + \boldsymbol{\omega}^{\mathcal{S}} \times I^{\mathcal{W}}\boldsymbol{\omega}^{\mathcal{W}} = -\tilde{\boldsymbol{T}}_{int}(\boldsymbol{p}^N, \dot{\boldsymbol{p}}^N, \ddot{\boldsymbol{p}}^N) \\
&\text{initial conditions:} \quad \boldsymbol{\omega}^{\mathcal{W}}(t_0) = \mathbf{0} \\
&\text{state path constraint:} \quad R(\boldsymbol{p}^N)^T\, \boldsymbol{\sigma}(t_0) \cdot \boldsymbol{\sigma}_s - \cos\alpha_s \leq 0, \\
&\text{control path constraints:} \quad \left\| \dot{\boldsymbol{H}}^{\mathcal{W}}(t) \right\|_\infty \leq \dot{H}^{\mathcal{W}}_{max} \\
&\qquad\qquad\qquad\qquad\quad \left\| \boldsymbol{\omega}^{\mathcal{W}/\mathcal{S}}(t) \right\|_\infty \leq \omega^{\mathcal{W}}_{max}
\end{aligned} \tag{5.61}$$

5.8.4 Numerical integration strategy

The problem is solved with the IPSO technique. However, the proposed IPSO-based algorithm applies only for Eq. (5.58a). In fact, Eq. (5.58b) must be integrated

in order to evaluate $\boldsymbol{\omega}^{\mathcal{W}}$ and $\dot{\boldsymbol{\omega}}^{\mathcal{W}}$, and consequently $\boldsymbol{H}^{\mathcal{W}}$ and $\dot{\boldsymbol{H}}^{\mathcal{W}}$. In this case, a fixed-step integrator is adopted. In particular, a 3(2) Runge-Kutta integration method based on the 3rd-order formula with a control given by the 2nd-order formula has been chosen. The Butcher tableau [120] that has been selected for this work is:

3(2) Butcher tableau

0			
1	1		
1/2	1/4	1/4	
	1/6	1/6	2/3
	1/2	1/2	

Referring to the 2nd- and 3rd-order approximations as y^{**} and y^{***}, respectively, the criterion to be satisfied by a successful integration at the step $n+1$ is:

$$\frac{\left\|y_{n+1}^{**} - y_{n+1}^{***}\right\|_{\infty}}{\max\left(\left\|y_n\right\|_{\infty}, 1.0\right)} < \varepsilon_r \tag{5.62}$$

where ε_r is the user-defined *relative* tolerance. The integration is carried out in the following way:

1. We try the numerical integration imposing a guess value of the integration time step given by $t_f/(L_0 - 1)$, where L_0 is the length of the time mesh used for the first integration attempt.

2. If the integration satisfies the imposed tolerance ε_r at every mesh point, then the process is finished.

3. If the integration violates the imposed tolerance at least at one point, the time step in reduced, the internal torques are interpolated on the updated time mesh and the process is repeated from point 2).

The mesh improvement is tried up to N_m times, so that at the nth integration trial the time mesh is nL_0. This procedure allows the use of a simple and effective integrator which usually requires a low computational effort (see the results section for details). However, if the 3(2) scheme fails to converge, a variable step Runge Kutta (4,5) is used (even though it increases the computational time).

The fundamental steps required by the proposed algorithm are reported in Algorithm 3. The convergence criterion is the one reported in Sec. 4.5.3.

Algorithm 3: IPSO algorithm

Initialization of constants, swarm and tolerances;
while $\delta J_g > \varepsilon_c$ **do**
  update the tolerances;
  reset $J_{p,i}^{(k)}, J_g^{(k)}, J_{l,i}^{(k)}$;
  while $N_{viol}^{(\boldsymbol{g}_{best})} > 0$ *and* $\delta J_g > \varepsilon_c$ **do**
    update w, c_l *and* c_g;
    for $j = 1 : N_{\mathcal{S}}$ **do**
      interpolate $\mathbf{p}$, $\dot{\mathbf{p}}$ *and* $\ddot{\mathbf{p}}$;
      compute the sensor rotation;
      compute of $\boldsymbol{\omega}$ *and* $\dot{\boldsymbol{\omega}}$, *and* $\tilde{\boldsymbol{T}}_{\boldsymbol{int}}$ *from Eq.* (5.58a);
      integrate the wheels dynamics, Eq. (5.58b);
      compute J_i^k;
      update $J_{p,i}^{(k)}, J_g^{(k)}, J_{l,i}^{(k)}$;
      update $N_{viol}^{(\boldsymbol{g}_{best})}$
    end
    for $j = 1 : N_{\mathcal{S}}$ **do**
      update the swarm velocity and position;
    end
  end
end

5.8.5 Constraint violation and performance index

In accordance with Sec. 4.5, the fitness function is selected in the form of an Exterior Penalty Function. The extended performance index is defined as in Eq. (5.63), i.e.

$$J^N = t_f + \pi P + C + \mu\, N_{viol} + f. \tag{5.63}$$

However, the control penalty function C must be revised to include the wheels' constraints. In further details, C is now defined as

$$C = \gamma_1 \sum_{j=1}^{3} \sum_{j=0}^{N_t} h_i(t_j) + \gamma_1 \sum_{j=1}^{3} \sum_{j=0}^{N_t} \varpi_i(t_j) \tag{5.64}$$

where γ_1 and γ_2 are user-defined weights and $h_i(t_i)$ and $\varpi_i(t_i)$ are

$$h_i(t_j) = \begin{cases} 0 & \text{if } \dot{H}_i^{\mathcal{W}}(t_j) - \dot{H}_{max}^{\mathcal{W}} < \Delta_{C_1} \\ 1 & \text{otherwise} \end{cases} \tag{5.65}$$

$$\varpi_i(t_j) = \begin{cases} 0 & \text{if } \omega_i^{\mathcal{W}}(t_j) - \omega_{max}^{\mathcal{W}} < \Delta_{C_2} \\ 1 & \text{otherwise} \end{cases} \tag{5.66}$$

being Δ_{C_1} and Δ_{C_2} are constraint tolerances. All the tolerances are implemented with the refined decreasing law defined in Sec. 4.5.2. The exit condition is the same as in Eq. (5.48).

5.8.6 Parameters and problem setting

The IPSO parameters used for the following numerical simulations are summarized in Table 5.12 and Table 5.13. The former reports the constant parameters, the latter takes into account the variable parameters, according to what has been described in the previous sections.

Having set $N_{\mathcal{P}} = 8$ in Table 5.12, the IPSO particle $\mathbf{x}$ (define as in Eq. (5.47)) contains 49 optimization variables. The constraint tolerances $\Delta_{\mathcal{C}_1}$ and $\Delta_{\mathcal{C}_2}$ in Table 5.13 are applied once the torques and the momenta of the wheels are normalized by $\dot{H}^{\mathcal{W}}_{max}$ and $H^{\mathcal{W}}_{max} = I^{\mathcal{W}}\omega^{\mathcal{W}}_{max}$, respectively. With regard to Δ_S and Δ_M, their initial value may be chosen in order to help the initial movement of the swarm, as will be explained later in this section. All results are obtained using a PC with a Intel® Core™ i7-2670QM CPU @ 2.20GHz processor and with 6.00 GB of RAM.

For the numerical simulations, the same low Earth-orbit satellite considered in Sec. 5.6 is taken into account. For the wheels, we assume that $\dot{H}^{\mathcal{W}}_{max} = 0.25$ Nm, $\omega^{\mathcal{W}}_{max} = 4000$ rpm and $I^{\mathcal{W}} = 0.1$ kg·m^2.

The geometries considered in this section are given in Table 5.14 for scenario 1, 2 and 3. For scenario 3, the two keep-out cones intersect and do not leave any free angles. As a consequence, the optimal maneuver will lie outside the keep-out cones.

The three geometries are reported in Fig. 5.21, where the graph is reported in the inertial reference system $\mathcal{B}_0$ in terms of latitude and longitude. In this case the reference system is chosen as to have the axes of the keep-out cones near the equator axis (where the latitude is equal to zero) in order to minimize the deformation of the cones. The sensor axis trajectories have been found using the IPSO. The computed maneuvers are completely feasible, i.e. they completely satisfy all the constraints in Eq. 5.61.

Tab. 5.12: IPSO constant parameters

Parameter	Value	Parameter	Value
$N_{\mathcal{S}}$	30	ε_c	1e-10
$N_{\mathcal{P}}$	8	ε_r	1e-4
N_δ	1500	γ_1	10
K	1000	γ_2	100
N_m	50	c_p	1.5
$\mathcal{D}$	7	μ	100
L_0	500	π	10

Tab. 5.13: IPSO variable parameters.

Parameter	Initial Value	Final Value
δ	0.05	1
$\Delta_{\mathcal{C}_1}$	0.25	0
$\Delta_{\mathcal{C}_2}$	0.25	0
Δ_S	variable	0
Δ_M	variable	0
w	1.2	0.6
c_l	2	0
c_g	0	2

Tab. 5.14: Sun and Moon direction for test scenarios with reaction wheels.

Scenario			Sun (in $\mathcal{B}_0$)			Moon (in $\mathcal{B}_0$)			Free Angle
N.	Θ_f (deg)	type	$\boldsymbol{\sigma}_{S,x}$	$\boldsymbol{\sigma}_{S,y}$	$\boldsymbol{\sigma}_{S,z}$	$\boldsymbol{\sigma}_{M,x}$	$\boldsymbol{\sigma}_{M,y}$	$\boldsymbol{\sigma}_{M,z}$	(deg)
1	60	roll	-0.58	-0.08	-0.81	0.41	-0.13	-0.91	0.52
2	60	roll	-0.65	-0.65	-0.40	0.10	-0.29	-0.95	0.84
3	60	roll	-0.58	-0.08	-0.81	0.29	-0.09	-0.95	0

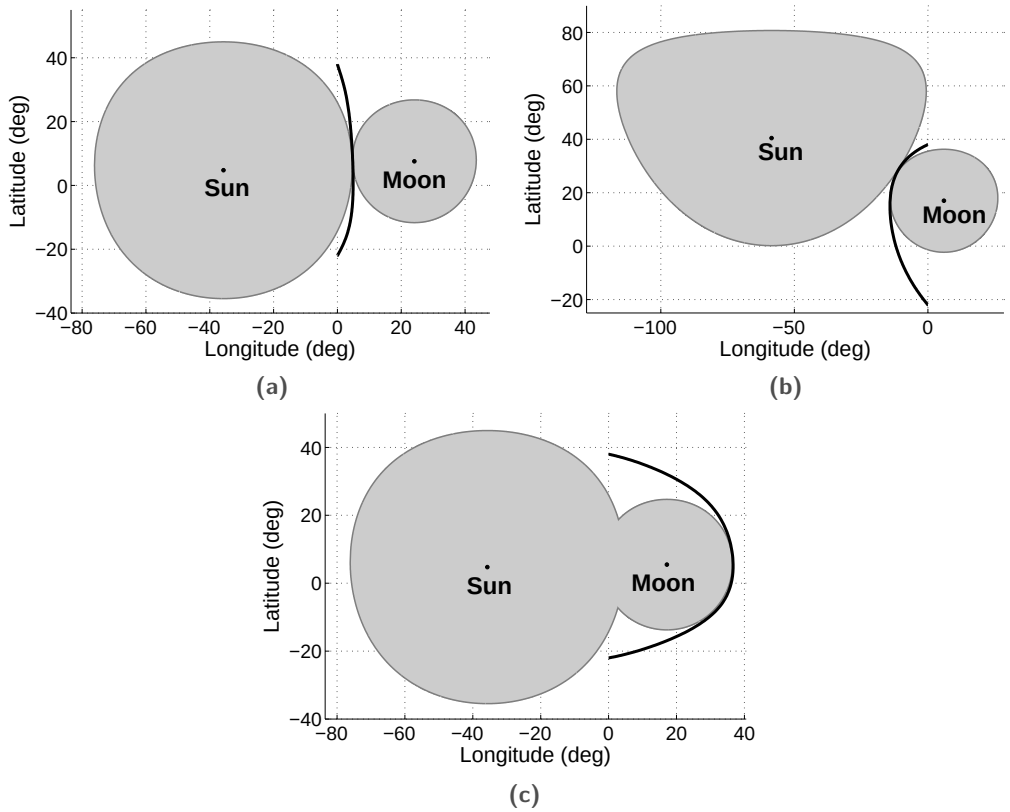


Fig. 5.21: IPSO maneuvers with rection wheels for scenario 1 (a), for scenario 2 (b) and for scenario 3 (c).

5.8.7 B-spline performance analysis

B-spline curves represent a very useful tool to approximate functions. The number of control points $N_{\mathcal{P}}$ and the degree $\mathcal{D}$ of the basis polynomials are the key parameters that can be chosen to maximize the performance of optimization algorithm. Consequently, a performance analysis has been carried out to find the best values to be used. Varying the value of $N_{\mathcal{P}}$ from 7 to 12, different polynomial degrees have been tried for scenario 1. For a clamped B-Spline, the maximum allowed degree is $\mathcal{D} = N_{\mathcal{P}} - 1$. For each test case, 200 simulations have been run. The average maneuver times are reported in Fig. 5.22. As it can be seen, the best results are

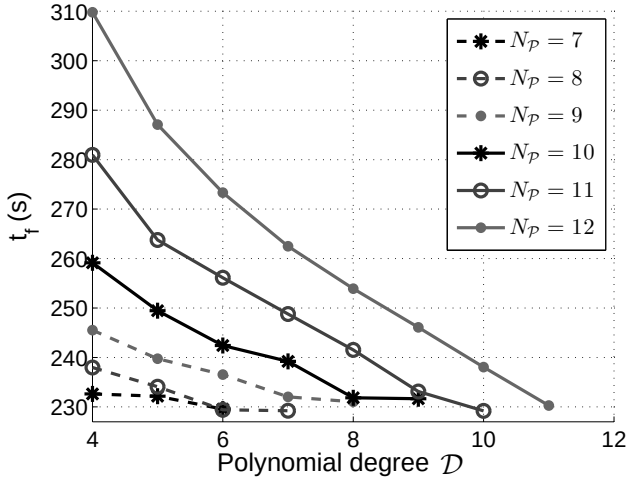


Fig. 5.22: Performance analysis for the definition of the B-Spline approximation.

always obtained when $\mathcal{D} = N_\mathcal{P} - 1$. The minimum maneuver time is achieved when $N_\mathcal{P} = 8$ and $N_\mathcal{P} = 11$. Consequently, $N_\mathcal{P} = 8$ and $\mathcal{D} = 7$ have been chosen in order to minimize the computational effort.

5.8.8 Active torque constraint

Detailed results of the IPSO solution for scenario 3 are reported through Fig. 5.23 and 5.24. The former reports the torques provided by the wheels and the related angular velocity history, while the latter reports the angular velocity history and the

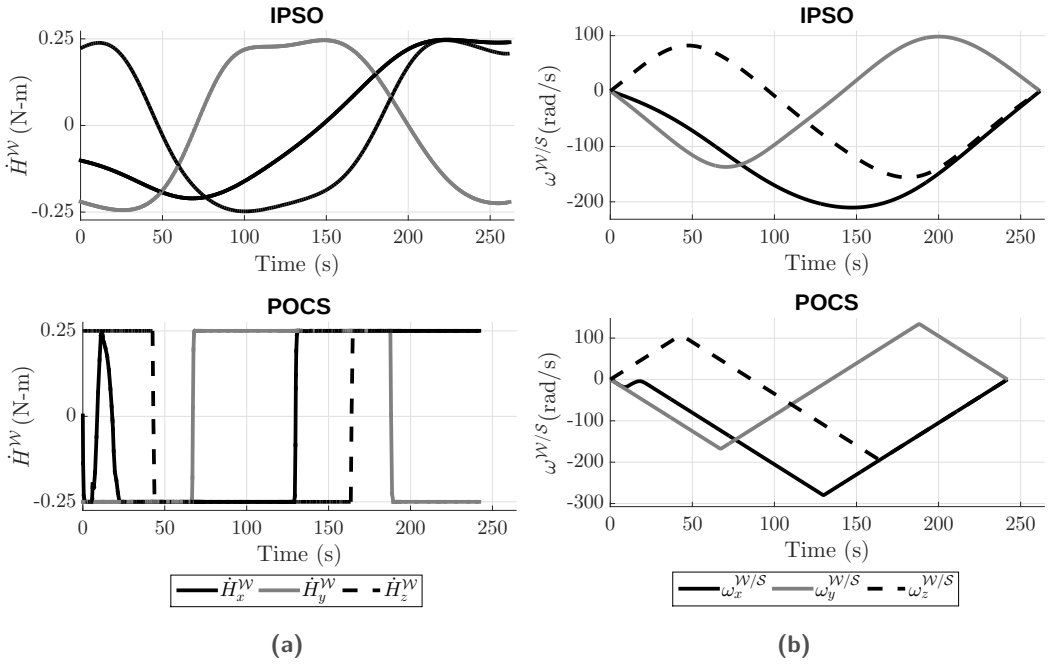


Fig. 5.23: Wheels torques (a) and angular velocities (b) from IPSO and POCS, scenario 2.

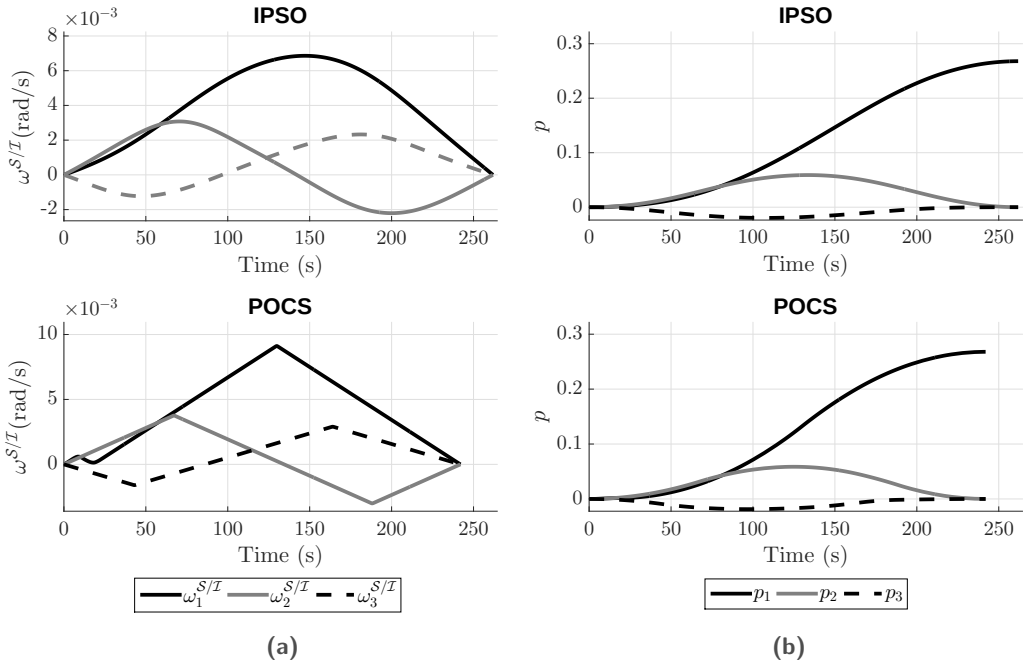


Fig. 5.24: Satellite angular velocity (a) and attitude (b) from IPSO and POCS, scenario 2.

attitude evolution of the satellite during the maneuver. These results are compared with a POCS [57] solution obtained with a mesh tolerance $\epsilon_m = 10^{-10}$. The maneuver time of the IPSO solution is 261.8 seconds whereas the POCS maneuver time is 242 seconds. As can be seen from the comparison, the IPSO is very close to the POCS solution. For example, looking at Fig. 5.23(a), it may be seen that the POCS bang-bang solution is well approximated from the IPSO. As a consequence, both the wheel kinematics in Fig. 5.23(b) and the kinematics of the satellite in Fig. 5.24 are quite the same using IPSO and POCS. The differences are due to the fact that the kinematics is parametrized with the B-splines using a fixed finite number of control points while POCS uses a polynomial interpolation based upon a time mesh which is continuously improved. As a consequence, the IPSO solution is a near-optimal solution, i.e. a completely feasible solution with a maneuver time very close to the minimum time found by POCS. Note that the wheel's angular velocity in Fig. 5.23(b) and the satellite angular velocity in Fig. 5.24(a) have a different order of magnitude which is consistent with the difference in the moment of inertia of wheels and satellite. Moreover, the sign of the angular velocity components for wheels and satellites are opposite consistently with Eq. (5.57).

As can be seen from the reported results, the torque provided by the wheels is saturated as a result of the minimum-time planning. On the contrary, initial conditions imposed in Eq. (5.61) do not lead to the saturation of the wheel's angular velocities, which remain well below the saturation limit of 400 rad/s through the slew maneuver. In Sec. 5.8.9 different initial conditions will be chosen such that the angular momentum constraint is active during the maneuver.

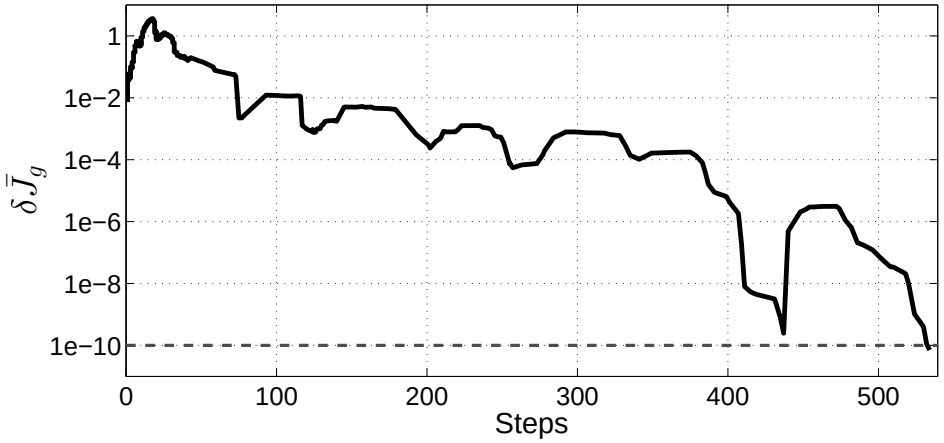


Fig. 5.25: Convergence index history, scenario 1.

With regard to the computational requirements for the obtainment of the solution, POCS may require long computational times depending on the required accuracy. For instance, when solving scenario 3 without an initial guess and $\epsilon_m = 10^{-10}$, POCS can require more than 5 million variables and, after more than 1 hour of computation, the computer was not able to solve the problem due to memory problems. On the contrary, the IPSO solution can be always found in a small amount of time: with the aforementioned computer, the computational time for the three presented cases is on average less than 150 seconds. It must be noted that other implementations are possible in order to decrease the computational time. For example, computational improvements may be achieved substituting the Matlab code with C or Fortran code, using ASICs or FPGAs instead of desktop computer (refer to [121] for more details). For scenario 1, a completely feasible maneuver is found with fewer than 55 iterations, on average.

Considering an IPSO solution for scenario 1, the behavior of the convergence index introduced in Eq. (5.48) is reported in Fig. 5.25. As can be seen, fewer than 600 iterations are required to obtain the desired convergence. A future implementation in C-code on ASIC or FPGA hardware will be carried out in order to verify if the computational effort of the proposed algorithm is consistent with the typical performances of satellite on-board hardware.

Finally, in Fig. 5.26 the IPSO results distribution over 1000 test cases is reported for the three problems considered. Since IPSO is an heuristic algorithm, it does not guarantee the same solution for the same initial conditions. However, the solutions are well distributed around maneuver times which are quite close to the POCS maneuver times (the maneuver times evaluated with POCS are 225.43 seconds for scenario 1, 241.98 second for scenario 2 and 388.15 seconds for scenario 3). The

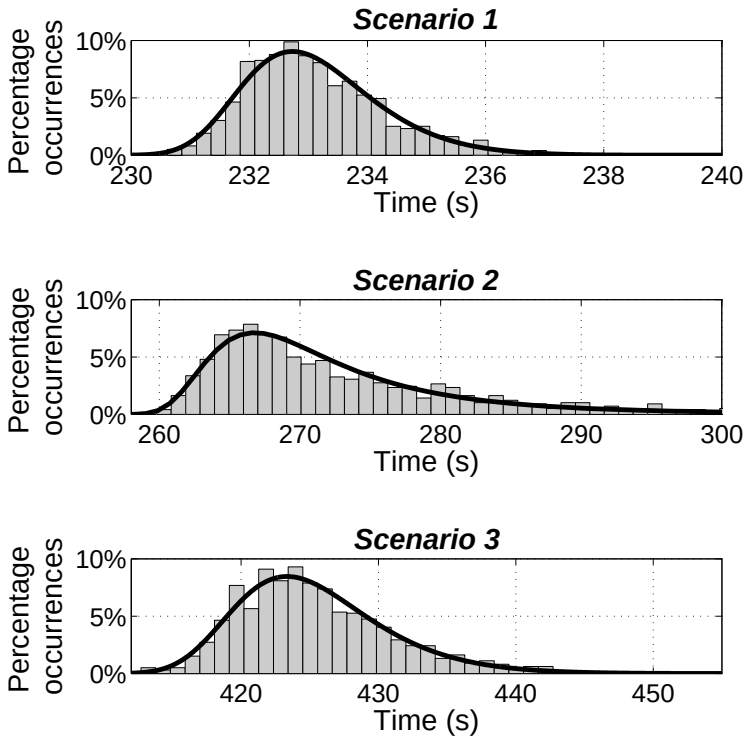


Fig. 5.26: IPSO results distribution over 1000 simulations for scenarios 1,2,3.

curves used to interpolate the histograms in Fig. 5.26 have been evaluated with a Generalized Extreme Value distribution defined by the parameters σ (shape), μ (location) and ζ (shape). For the three studied cases, these parameters get the following values:

- $[\sigma, \mu, \zeta] = [-0.1047, 1.0108, 2.3262e2]$ for scenario 1,
- $[\sigma, \mu, \zeta] = [0.2559, 5.4685, 2.6796e + 02]$ for scenario 2,
- $[\sigma, \mu, \zeta] = [-0.0789, 4.6751, 4.2296e + 02]$ for scenario 3.

It is worth noting that on the one hand scenario 3 cannot be solved using POCS without an initial guess solution, on the other hand the same problem can be solved using the IPSO solution as the best guess for the POCS. In this case, the performances of POCS are improved and the solution may be found in about 200 seconds.

5.8.9 Active momentum constraint

In the previous section the IPSO approach has been tested and verified with different scenarios. The control constraint was active whereas the angular momentum constraint was inactive.

In this section the optimal control problem statement in Eq.(9.5) is slightly modified as a nonzero initial value of the wheels' angular velocity is imposed to reach the angular momentum saturation during the slew maneuver. Choosing the initial velocities of the three wheels as

$$\boldsymbol{\omega}^{\mathcal{W}}(t_0) = [-300,\, +200,\, -100]^T \text{ rad/s}, \tag{5.67}$$

the results of scenario 2 are reported in Fig. 5.27. In Fig. 5.27(a), the torques provided by the wheels are shown, whereas the wheels' angular velocities are reported in Fig. 5.27(b). The angular velocities and the attitude history of the satellite are reported in Fig. 5.27(c)-(d). Comparing these results with Fig. 5.23, it can be seen that different initial conditions of the wheels affect the result of the planning algorithm. For instance, $\dot{H}_x^{\mathcal{W}}$ in Fig. 5.23(a) is bang-bang, whereas $\dot{H}_x^{\mathcal{W}}$ in Fig. 5.27(a) is bang-off-bang. In fact, the torque is constrained to go to zero since the wheel's velocity saturation has been reached as one can see looking at $\omega_x^{\mathcal{W}/\mathcal{S}}$ in Fig. 5.27(b). As a consequence, the angular momentum saturation increases the maneuver time, which is now 341.4 seconds.

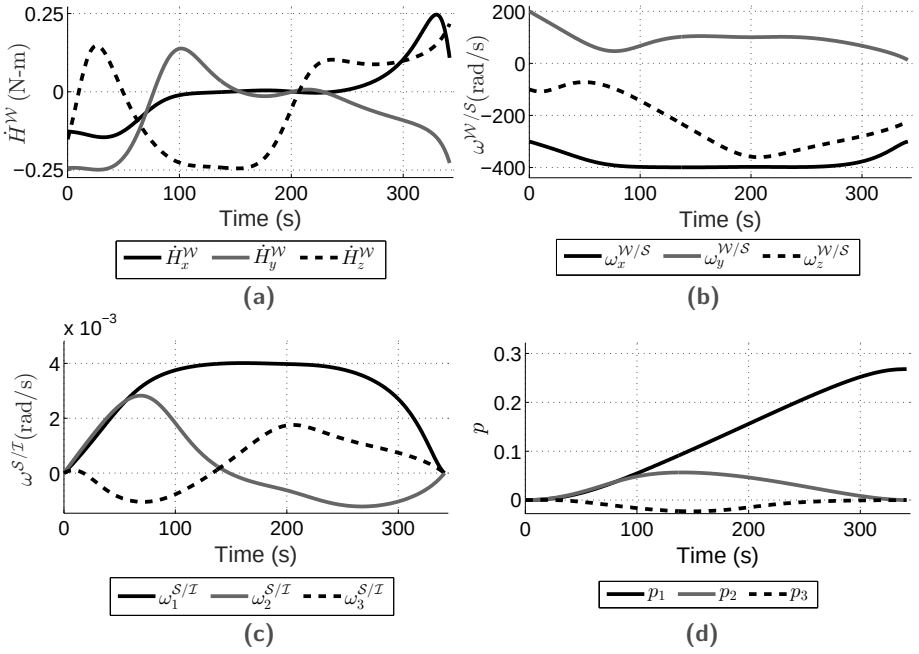


Fig. 5.27: Results with wheels' angular velocity saturation, scenario 2.

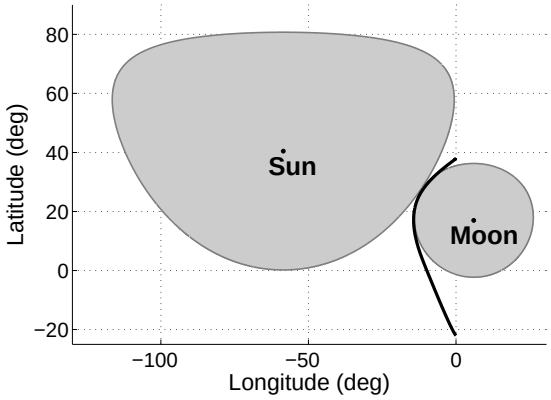


Fig. 5.28: Sensor axis trajectory with wheels' angular velocity saturation, scenario 2.

The optical sensor trajectory is shown in Fig. 5.28. As a result of the angular momentum saturation, the trajectory is slightly different from the one reported in Fig. 5.21(b).

5.9 Endnotes

In this chapter it has been shown that the Particle Swarm Optimization may be used for planning near-optimal constrained maneuvers. When the proposed Inverse-dynamics Particle Swarm Optimization is used, the boundary and the path constraints are fully satisfied. The direct and inverse dynamics approaches have been described and the advantages of the latter with respect to the former have been defined. Accordingly, only the first problem has been solved with the direct approach, while all the other simulations have been carried out with the inverse technique (as it will be done for the majority of the following chapters)

Different satellite models have been considered. First, external torques have been considered, and the planning problem has been solved considering end-point and path constraints (the keep-out cone constraint). Second, the reaction wheels dynamics has been included in the optimization problem. In this case, a hybrid method mixing the inverse-dynamics particle swarm optimization and the numerical integration has been shown. In both cases, the Inverse Method guarantees near minimum-time solutions that fully satisfy boundary constraints, path constraints and control constraints. The presented method always guarantees a feasible solution with a reduced computational effort with regard to the pseudospectral approach, even when the latter is not able to find a solution. In fact, using the inverse dynamics approach and the proper setting of the particle swarm parameters may lead to errors in the final maneuver time below 1% with respect to a reference pseudospectral method. Moreover, it was underlined that both the results and the computational

time do not change considering the different geometries of the keep-out cones or different characteristics of the maneuver. On the contrary, the calculation of the solution through an optimization solver such as a pseudospectral optimization software may require very high computational times. In our tests, the computational time required by the Inverse Method is up to $1/15$ of the time required by the pseudospectral optimization software without initial guess.

The low computational effort and the satisfaction of all the imposed constraints make the proposed approach suitable in the perspective of achieving fully autonomous satellites.

6

Inverse-Dynamics Particle Swarm Optimization applied to General Bolza Problems*

Abstract

So far, the Inverse-dynamics Particle Swarm Optimization has been successfully applied to minimum-time problems. The advantages of this technique, formulated with the differentially flat approach, lie in the global search ability of the optimizer and the reduction of the independent functions due to the exploitation of the differential flatness. However, it is known that optimal control problems formulated with either differential inclusion or differential flatness can lead to nonconvex problems with undesirable numerical properties. This chapter is intended to show that, considering difficult problems with nonconvex state constraints and nonconvex cost functions, the proposed numerical technique can lead to feasible near-optimal solutions. Minimum-time, minimum-energy and minimum-effort maneuvers are addressed considering the constrained slew maneuver as a test case.

Nomenclature

$\mathbf{x}$	=	PSO particle	$\boldsymbol{I}$	=	Inertia tensor
$\boldsymbol{\omega}$	=	Body angular velocity	t	=	Time (s)
$\boldsymbol{u}$	=	External control	J	=	Performance index
$(\cdot)^N$	=	Numerical approximation	$\dot{(\cdot)}$	=	First time derivative
$\ddot{(\cdot)}$	=	Second time derivative	$\boldsymbol{p}$	=	Modified Rodrigues Parameters
$N_{\mathcal{P}}$	=	Number of approximation coefficients	$(\cdot)_{0/f}$	=	Initial/final time value
$\boldsymbol{\sigma}$	=	Optical axis	$\boldsymbol{\sigma}_s$	=	Light-source direction
α_s	=	Light-source half-angle (rad)	β	=	Angle between $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}_s$ (rad)
$N_{\mathcal{P}}$	=	Number of approximation coefficients	N_T	=	Number of discretization points
Ψ	=	Angular kinematics matrix	R	=	Rotation matrix
$\tilde{(\cdot)}$	=	Approximation parameter	u_{max}	=	Maximum external control
$\mathbb{X}$	=	State space	$\mathbb{U}$	=	Control space
$\mathbb{Y}$	=	Flat output space	Θ_f	=	Slew angle
E	=	End-point cost functional	F	=	Running cost functional
$\mathbf{b}$	=	Boundary constraint function	$\mathbf{p}$	=	Path constraint function

*This chapter is based on Ref. [122].

6.1 Introduction

In Chapter 2 several different ways to set and transcribe an optimal control problem have been described. Usually, the optimal control theory typically employed in engineering fields is based upon the state and the control functions. The differential inclusion is another formulation that allows one to include the differential constraint inside the optimization problem. Finally, the differential flatness formulation is based on the identification of a minimum number of independent flat outputs that can completely describe and solve an optimal control problem.

It has been already said that, for generic Bolza problems with running cost and terminal cost, the differential flatness formulation may transform an initial convex cost functional into a nonconvex one. Since the convexity problem plays a crucial role both for the mathematical treatment of OCPs and for numerical applications [35], this chapter is intended to study the behavior of the IPSO when different cost functional are taken into account. The goal of this work is to show that such a technique can successfully manage complex noncovex problems. However, reporting only an example of application, this work is far from assuring that the IPSO can manage all problems within the class of the nonconvex OCPs.

This chapter is organized as follows. First, in Sec. 6.2 the example optimization problem is recalled. Three different formulations of an OCP are then reported, where the last one is the employed differential flatness one. In Sec. 6.3 the influence of the cost function on the convexity of the problem is investigated and the main features of the proposed numerical approach are described. Finally, results are reported in Sec. 6.4 and conclusions are given in Sec. 6.5.

6.2 Formulation of the optimization problem

Let us recall what has been stated in Chapter 2. Three different possible ways to set and solve a Bolza Optimal Control Problem (BOCP) have been described, i.e. the state-control space formulation, the differential inclusion formulation and the differential flatness approach.

The problem to be solved is the same described in Sec. 5.2, i.e. the constrained reorientation maneuver of a satellite. The IPSO technique is used, and consequently the differential flatness implementation described in Chapter 4 is employed.

It is well-known that the numerical complexity of an optimal control problem is due not only to the number of unknown parameters but also to the characteristics related to the chosen parametrization of the problem. In further detail, an optimization

problem may have a convex cost function with one representation and a nonconvex cost function with another one. This is what usually happens when the differentially flat approach is employed.[34, 47]

First of all, let us recall some results described in Chapter 5. It has been shown that the external control $\boldsymbol{u}$ for the satellite slew maneuver may be written as

$$\boldsymbol{u} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}. \tag{6.1}$$

In fact, recalling what has been already stated in Chapter 5, the angular velocity $\boldsymbol{\omega}$ and the angular acceleration $\dot{\boldsymbol{\omega}}$ are functions of the modified Rodrigues parameters $\boldsymbol{p}$ and their time derivatives, i.e.

$$\boldsymbol{\omega} = 4\Psi^{-1}\dot{\boldsymbol{p}}, \tag{6.2}$$

$$\dot{\boldsymbol{\omega}} = 4\left(\dot{\Psi}^{-1}\,\dot{\boldsymbol{p}} + \Psi^{-1}\,\ddot{\boldsymbol{p}}\right), \tag{6.3}$$

where

$$\Psi = \left[\left(1 - \boldsymbol{p}^T\boldsymbol{p}\right) I + 2[\tilde{\boldsymbol{p}}] + 2\boldsymbol{p}\boldsymbol{p}^T\right], \quad \Psi^{-1} = \frac{\Psi^T(\boldsymbol{p})}{(1+\boldsymbol{p}^T\boldsymbol{p})^2}, \tag{6.4}$$

$$\dot{\Psi} = \left[-(\dot{\boldsymbol{p}}^T\boldsymbol{p} + \boldsymbol{p}^T\dot{\boldsymbol{p}})I + 2[\tilde{\dot{\boldsymbol{p}}}] + 2(\dot{\boldsymbol{p}}\boldsymbol{p}^T + \boldsymbol{p}\dot{\boldsymbol{p}}^T)\right], \tag{6.5}$$

$$\dot{\Psi}^{-1} = \frac{\dot{\Psi}^T}{(1+\boldsymbol{p}^T\boldsymbol{p})^2} - \frac{2\Psi^T}{(1+\boldsymbol{p}^T\boldsymbol{p})^3}(\dot{\boldsymbol{p}}^T\boldsymbol{p} + \boldsymbol{p}^T\dot{\boldsymbol{p}}). \tag{6.6}$$

Let us also recall that the sensor axis $\boldsymbol{\sigma}$ orientation during the maneuver is given as

$$\boldsymbol{\sigma}(t) = R(\boldsymbol{p})^T\,\boldsymbol{\sigma}(t_0). \tag{6.7}$$

where R is evaluated with Eq. (5.25). Moreover, the keep-out cone constraint is defined as

$$C_s(t) = \boldsymbol{\sigma}(t)\cdot\boldsymbol{\sigma}_s - \cos(\alpha_s) \leq 0 \qquad \forall t \in [t_0, t_f], \tag{6.8}$$

where $\boldsymbol{\sigma}_s$ and α_s are the axis and the half-angle of the light source.

In the next section, a summary of the three different problem formulations introduced in Chapter 2 will be reported. Moreover, the the three formulation will be applied to the slew maneuver problem to underline the effect of the complexity of the transcribed problem.

6.2.1 Problem $\mathrm{P_{SC}}$

Let $\boldsymbol{x}(\cdot)$ be the state function with values $\boldsymbol{x}(t) \in \mathbb{X} \subset \mathbb{R}^{N_x}$, where N_x is the dimension of the state space, and let $\boldsymbol{u}(\cdot)$ be the control function with values $\boldsymbol{u}(t) \in \mathbb{U} \subset \mathbb{R}^{N_u}$, where N_u is the dimension of the control space. Following the

standard state-control space formulation, a Bolza problem is an optimization problem usually defined as

$$\begin{aligned} &minimize \\ J[\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot), t_f] &= E(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} F(\boldsymbol{x}(t), \boldsymbol{u}(t), t) dt \end{aligned} \tag{6.9}$$

subject to boundary constraints $\mathbf{b}$ and path constraints $\mathbf{p}$

$$\mathbf{b}(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_f) = \mathbf{0}, \tag{6.10}$$

$$\mathbf{p}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \leq \mathbf{0} \qquad \forall t \in [t_0, t_f]. \tag{6.11}$$

Note that Eq. (6.11) imposes inequality constraints on both state and control. In Eq. (6.9), $E \colon \mathbb{R}^{N_x} \times \mathbb{R}^{N_x} \times \mathbb{R} \mapsto \mathbb{R}$ is the terminal cost function, $F \colon \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R} \mapsto \mathbb{R}$ is the running cost function and

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \quad \text{a.e. on } [t_0, t_f]. \tag{6.12}$$

For the problem described in the previous section, $N_x = 6$, $N_u = 3$ and $\boldsymbol{x} = [\boldsymbol{p}\,; \boldsymbol{\omega}]$. Eq. (6.12) is specified as

$$\begin{bmatrix} \dot{\boldsymbol{p}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} 0.25\, \Psi(\boldsymbol{p}) \boldsymbol{\omega} \\ I^{-1} \left(\boldsymbol{u} - \boldsymbol{\omega} \times I\boldsymbol{\omega} \right) \end{bmatrix}. \tag{6.13}$$

Boundary constraints are

$$\boldsymbol{b}_1 = \begin{bmatrix} \boldsymbol{p}(t_0) - \boldsymbol{p}_0 \\ \boldsymbol{\omega}(t_0) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{b}_2 = \begin{bmatrix} \boldsymbol{p}(t_f) - \boldsymbol{p}_f \\ \boldsymbol{\omega}(t_f) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{6.14}$$

whereas the path constraints are

$$p_1 = R(\boldsymbol{p}(t))^T \boldsymbol{\sigma}(t_0) \cdot \boldsymbol{\sigma}_s - \cos(\alpha_s) \leq 0 \tag{6.15}$$

$$p_2 = \|\boldsymbol{u}(t)\|_\infty - u_{max} \leq 0 \quad \forall t \in [t_0, t_f]. \tag{6.16}$$

The L_∞-norm is defined as $\|\boldsymbol{u}(t)\|_\infty = \max\{|u_i(t)| : i = 1, 2, 3\}$ where $u_i(t)$ is the ith component of the control $\boldsymbol{u}(t)$. Note that $\boldsymbol{b}_1$, $\boldsymbol{b}_2$ and p_2 are convex constraints, whereas p_1 is nonconvex.

From the theoretical point of view, when solving a fixed-time problem, the total number of unknowns is given by the N_x components of the state function plus the N_u components of the control functions. For a free-time problem, instead, one more value must be determined corresponding to the total time of application of the control.

Employing a pseudospectral approach, the number of optimization parameters would be $(N_x + N_u)N_{\mathcal{P}} = 9N_{\mathcal{P}}$ for a fixed-time problem and $9N_{\mathcal{P}} + 1$ for a free-time problem. Using a direct dynamics approach, the number of optimization parameters is reduced to $3N_{\mathcal{P}}$ for a fixed-time problem and $3N_{\mathcal{P}} + 1$ for a free-time problem.

6.2.2 Problem $\mathrm{P_{DI}}$

In this case, the control $\boldsymbol{u}$ does not appear explicitly in the cost function, i.e. the optimal control problem is written as

$$\begin{aligned} &minimize \\ J[\boldsymbol{x}(\cdot), t_f] &= E(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} F\left(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t\right) dt \end{aligned} \tag{6.17}$$

over all arcs $\boldsymbol{x}(\cdot)$ satisfying the differential inclusion

$$\dot{\boldsymbol{x}}(t) \in \boldsymbol{\mathcal{F}}(\boldsymbol{x}(t)) \quad \text{a.e. on } [t_0, t_f] \tag{6.18}$$

where

$$\boldsymbol{\mathcal{F}}(\boldsymbol{x}(t)) = \{\boldsymbol{v} \in \boldsymbol{\mathcal{X}} : \boldsymbol{v} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)),\ \boldsymbol{u}(t) \in \mathbb{U}(t)\} \tag{6.19}$$

Boundary and path constraints are imposed as

$$\mathbf{b}(\boldsymbol{x}(t_0), \boldsymbol{x}(t_f), t_f) = \mathbf{0}, \tag{6.20}$$

$$\mathbf{p}(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), t) \leq \mathbf{0} \qquad \forall t \in [t_0, t_f]. \tag{6.21}$$

For the proposed problem, a solution can be obtained by means of the differential inclusion approach approximating only the state and writing $\boldsymbol{u}$ as a function of the state, i.e.

$$\boldsymbol{u} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}. \tag{6.22}$$

Eq. (6.22) is employed to write the path constraints and the cost function as a function of the state (see [47] for other examples and details). Boundary constraints are expressed in the same way as in Eq. (6.14), as well as p_1 remaining unchanged. On the contrary, the path constraint p_2 is given by

$$p_2 = |I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega}\|_\infty - u_{max} \leq 0. \tag{6.23}$$

Note that, while p_2 in Eq. (6.16) was a convex constraint, now p_2 is nonconvex. Choosing to approximate all the N_x components of the state function, the total number of unknowns is equal to $N_x N_{\mathcal{P}} = 6N_{\mathcal{P}}$ when solving a fixed-time problem whereas, for a free-time problem, it is $6N_{\mathcal{P}} + 1$.

6.2.3 Problem $\mathrm{P_{DF}}$

Let $\boldsymbol{y}(t) \in \mathbf{R}^{N_u}$ be the flat output defined as [32]

$$\boldsymbol{y} = \boldsymbol{c}\left(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, ..., \boldsymbol{u}^{(\alpha)}\right), \quad \boldsymbol{y} \in \mathbb{Y} \subset R^{N_u}, \tag{6.24}$$

for some $\alpha \in \mathbb{N}$. State and the control are written as

$$\boldsymbol{x} = \boldsymbol{a}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, ..., \boldsymbol{y}^{(\beta)}\right), \quad \boldsymbol{u} = \boldsymbol{b}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, ..., \boldsymbol{y}^{(\beta+1)}\right) \tag{6.25}$$

Using the notation introduced in Sec. 2.6, the Bolza problem can be stated as a function of the flat output as

$$\begin{gathered} minimize \\ J[\boldsymbol{y}(\cdot), t_f] = E(\boldsymbol{y}(t_0), \boldsymbol{y}(t_f), t_f) + \int_{t_0}^{t_f} F\left(\boldsymbol{y}(t), \{\boldsymbol{y}^{(\beta)}(t)\}, t\right) dt. \end{gathered} \tag{6.26}$$

Boundary and path constraints are imposed as

$$\mathbf{b}(\boldsymbol{y}(t_0), \{\boldsymbol{y}^{(\beta)}(t_0)\}, \boldsymbol{y}(t_f), \{\boldsymbol{y}^{(\beta)}(t_f)\}, t_f) \leq \mathbf{0}, \tag{6.27}$$

$$\mathbf{p}(\boldsymbol{y}(t), \{\boldsymbol{y}^{(\beta+1)}(t)\}, t) \leq \mathbf{0}. \tag{6.28}$$

In Chapter 5, it has already been shown that the flat output may be chosen as the vector $\boldsymbol{p}$. In fact, from the relationships reported through Eqs. (6.2)-(6.6), the state can be written as a function of $\boldsymbol{p}$ and $\dot{\boldsymbol{p}}$, i.e.

$$\boldsymbol{x} = \boldsymbol{a}\left(\boldsymbol{p}, \dot{\boldsymbol{p}}\right) = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} I_{3\times 3} \\ 4\Psi^{-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ \dot{\boldsymbol{p}} \end{bmatrix}. \tag{6.29}$$

With regard to the control, it can be easily written as a function of the flat parameter, i.e.

$$\boldsymbol{u} = \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \boldsymbol{b}(\boldsymbol{p}(t), \dot{\boldsymbol{p}}(t), \ddot{\boldsymbol{p}}(t)). \tag{6.30}$$

where the function $\boldsymbol{b}(\boldsymbol{p}(t), \dot{\boldsymbol{p}}(t), \ddot{\boldsymbol{p}}(t))$ is evaluated by means of Eq. (6.4) and Eq. (6.6),

$$\boldsymbol{b}(\boldsymbol{p}(t), \dot{\boldsymbol{p}}(t), \ddot{\boldsymbol{p}}(t)) = 4I\left(\dot{\Psi}^{-1}(\boldsymbol{p})\,\dot{\boldsymbol{p}} + \Psi^{-1}(\boldsymbol{p})\,\ddot{\boldsymbol{p}}\right) + 16\left(\Psi^{-1}(\boldsymbol{p})\dot{\boldsymbol{p}}\right) \times I\left(\Psi^{-1}(\boldsymbol{p})\dot{\boldsymbol{p}}\right). \tag{6.31}$$

Accordingly, thanks to Eq. (5.33), the boundary constraints are now dependent only of the MRPs as

$$\mathbf{b}_1 = \begin{bmatrix} \boldsymbol{p}(t_0) - \boldsymbol{p}_0 \\ \dot{\boldsymbol{p}}(t_0) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} \boldsymbol{p}(t_f) - \boldsymbol{p}_f \\ \dot{\boldsymbol{p}}(t_f) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{6.32}$$

and the path constraints take the following form

$$p_1 = R(\boldsymbol{p}(t))^T \, \boldsymbol{\sigma}(t_0) \cdot \boldsymbol{\sigma}_s - \cos(\alpha_s) \leq 0, \tag{6.33}$$

$$p_2 = \left\| 4I \left(\dot{\Psi}^{-1}(\boldsymbol{p}) \, \dot{\boldsymbol{p}} + \Psi^{-1}(\boldsymbol{p}) \, \ddot{\boldsymbol{p}} \right) + 16 \left(\Psi^{-1}(\boldsymbol{p}) \dot{\boldsymbol{p}} \right) \times I \left(\Psi^{-1}(\boldsymbol{p}) \dot{\boldsymbol{p}} \right) \right\|_\infty - u_{max} \leq 0. \tag{6.34}$$

This approach leads to the minimum number of unknown parameters without requiring a numerical integration ($3N_\mathcal{P}$ for a fixed-time problem, $3N_\mathcal{P} + 1$ for a free time problem). Note that $\dot{\boldsymbol{p}}$ and $\ddot{\boldsymbol{p}}$ are not independent functions as they are derived analytically from $\boldsymbol{p}$ once the specific polynomial approximation is imposed.

The issue often related to this approach is that it can add some difficulties to the cost function as described in the next section. Let us consider the improved B-spline approximation defined in Sec. 4.2. Eq. (6.30) can be rewritten after having performed the numerical transcription as

$$\boldsymbol{u}^N = \boldsymbol{b}(\boldsymbol{p}^N(t), \dot{\boldsymbol{p}}^N(t), \ddot{\boldsymbol{p}}^N(t)) = \tilde{\boldsymbol{b}}(\tilde{\boldsymbol{p}}_1, ..., \tilde{\boldsymbol{p}}_{N_u}, \tilde{\boldsymbol{t}}_1, ..., \tilde{\boldsymbol{t}}_{N_u}, t_f), \tag{6.35}$$

where $\tilde{\boldsymbol{p}}_i$ and $\tilde{\boldsymbol{t}}_i$ are the parameters defining the basic B-spline approximation and entering into the generic PSO particle, as defined in Sec. 5.4. As can be seen, the function $\tilde{\boldsymbol{b}}$ is nonlinear and nonconvex (this is easy to understand as the cross product in Euler's equation leads to sinusoidal nonconvex functions). The same nonconvexity issue would appear when transcribing the example problem defined in Sec. 2.3 and using the differential flatness formulation.

6.3 Cost function influence on the solution approaches

Let us analyze three different type of Bolza problem to determine how the transcription can affect the mathematical properties of the cost function. The reader can refer to Sec. 4.5 for further details.

Minimum-time maneuvers In this case the Bolza cost functional is expressed as a Mayer cost functional, i.e.

$$J = t_f \tag{6.36}$$

As can be seen, this cost functional is not affected by the parameterizations of the problem presented before. In fact, neither the state or the control affect J. Consequently, it is expected that the differential flatness formulation is the best approach to solve the problem.

For the problem accounted for in this chapter, the PMP assures that the optimal control policy is given by a bang-bang structure [29].

Minimum-effort maneuvers Using the formulation of problem $\mathrm{P_{SC}}$, the cost functional is expressed as

$$J = \int_{t_0}^{t_f} \sum_{i=1}^{N_u} |u_i(t)|\, dt \tag{6.37}$$

where $u_i(t)$ is the ith component of the vector $\boldsymbol{u}(t)$. For the proposed problem, it is clear that both the differential inclusion and the differential flatness approaches modify an original convex problem (in the state-control space) into a nonconvex one. In fact, $\boldsymbol{u}$ is given by a cross product in the differential inclusion formulation. Moreover, the non-linear relationships in Eq. (6.2) and Eq. (6.3) must be considered for the differential flatness formulation. Such nonconvexity problems may increase the difficulties of finding numerical solutions to the optimization problem when using collocation-based approaches.[34].

Minimum-energy maneuvers Using the formulation of problem $\mathrm{P_{SC}}$, the cost functional is expressed as

$$J = \int_{t_0}^{t_f} \sum_{i=1}^{N_u} u_i(t)^2 dt. \tag{6.38}$$

This example is similar to the previous one. In fact, the same nonconvexity issues are introduced with the differential inclusion and the differential flatness formulations.

As a result, from the previous discussions it appears that the differential flatness formulation is advantageous with respect to the reduction of the independent unknowns but disadvantageous as it introduces additional nonconvexity issues to the optimization problem.

Nonconvexity is usually related to the presence of several local minima which often makes it difficult finding the global optimal solution for a gradient-based optimizer. However, heuristic techniques used in the last decades may succeed in finding the global optimum in such scenarios. Accordingly, in the following section

we will consider the results obtained with the IPSO technique, verifying its ability to solve nonconvex OCPs.

6.4 Results

Results are obtained using the improved IPSO scheme described in Chapter 4. The performance index and the penalty functions are defined as in Chapter 5. The IPSO parameters used for the following numerical simulations are the same reported in Table 5.12 and Table 5.13. Applying the improved B-Spline approximation to the three components of the MRPs and introducing the related time mesh, the IPSO particle is defined as

$$\mathbf{x} = [\tilde{\boldsymbol{p}}_1, \tilde{\boldsymbol{p}}_2, \tilde{\boldsymbol{p}}_3, \tilde{\boldsymbol{t}}_1, \tilde{\boldsymbol{t}}_2, \tilde{\boldsymbol{t}}_3, t_f] \in \mathbb{R}^{6N_{\mathcal{P}}+1} \tag{6.39}$$

where t_f is the maneuver time which must be taken into account only for minimum-time maneuvers. Having set $N_{\mathcal{P}} = 8$ in Table 7.2, the optimization variables within the IPSO particle $\mathbf{x}$ in Eq. (7.5) are 49 for a free-time OCP and 48 for a fixed-time OCP. The keep-out cone constraint tolerances are $\Delta_{1,2}$, whereas the control tolerance Δ_3 is equal to zero as this constraint is easily satisfied.

Results have been computed normalizing the external control by $\tau_u = u_{max}$, the time by $\tau_t = 1/\sqrt{I_x/u_{max}}$ and the angular velocity by $\tau_\omega = \tau_t^{-1}$.

6.4.1 Validation of the algorithm without cone constraints

First, let us consider a slew maneuver with $\Theta_f = 45$ deg without path constraints accomplished by an inertially symmetric satellite. This case has been already considered in the literature[123] for minimum-time maneuvers. For this simple case the problem is reduced to a double integrator for the three body axes. The expected and the obtained results are listed below:

- For a minimum-time maneuver, the optimal control evaluated with the PMP is bang-bang[25]. The control policy obtained with the IPSO is reported in Fig. 6.1.(a) whereas the angular velocity and the attitude histories are reported in Fig. 6.2.(a) and Fig. 6.3.(a), respectively. The results represent a nice approximation of the optimal results as the final time is $1.776\,\tau_t$ with a relative error equal to 0.0147 with respect to the known optimal time[123]. This error is consistent with the inverse dynamics technique[1].

- For a minimum-effort maneuver, the optimal control is bang-off-bang[25]. The control policy obtained with the IPSO imposing a maneuver time of 10 τ_t is

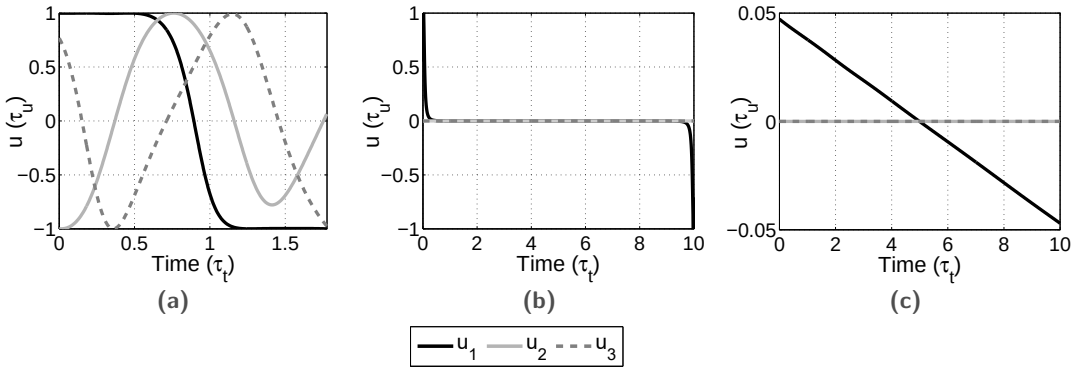


Fig. 6.1: Control policy for minimum-time (a), minimum-effort (b) and minimum-energy (c) maneuvers without cone constraint.

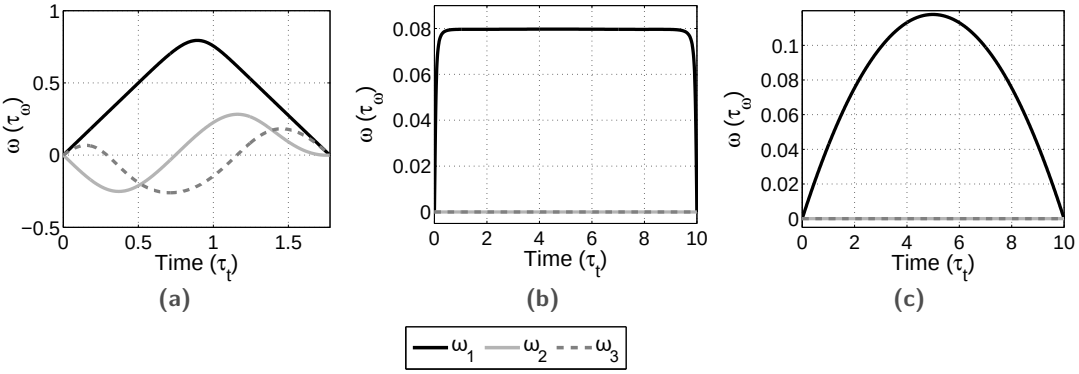


Fig. 6.2: Angular velocity histories for minimum-time (a), minimum-effort (b) and minimum-energy (c) maneuvers without cone constraint.

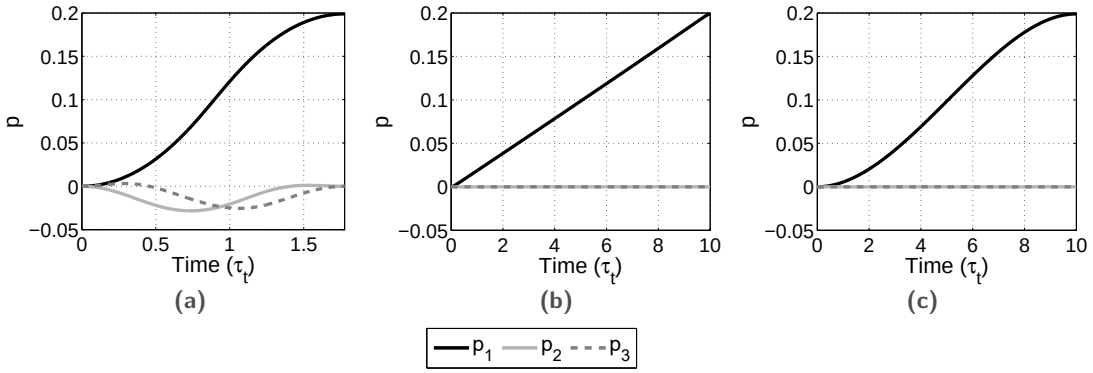


Fig. 6.3: Attitude histories for minimum-time (a), minimum-effort (b) and minimum-energy (c) maneuvers without cone constraint.

reported in Fig. 6.1.(b) whereas the angular velocity and the attitude histories are reported in Fig. 6.2.(b) and Fig. 6.3.(b), respectively. As can be seen, the control along $\boldsymbol{x}_{\mathcal{B}}$ is a bang-off-bang, whereas there is no motion along the other two axes.

- For a minimum-energy maneuver, the optimal control is linear[25]. The control policy obtained with the IPSO imposing a maneuver time of 10 τ_t is reported in Fig. 6.1.(c) whereas the angular velocity and the attitude histories are reported in Fig. 6.2.(c) and Fig. 6.3.(c), respectively. The results are consistent with the theory, as for the dynamical systems as the one considered here the control is linear along $\boldsymbol{x}_{\mathcal{B}}$ and equal to zero along $\boldsymbol{y}_{\mathcal{B}}$ and $\boldsymbol{z}_{\mathcal{B}}$.

6.4.2 Results with cone constraints

Let us consider test case 1 from Sec. 5.6. Results are reported in Fig. 6.4, Fig. 6.5 and Fig. 6.6 for the control policy, the angular velocity and the attitude history, respectively. For the minimum-effort and minimum-energy cases, the maneuver time has been fixed to 10 τ_t. As can be seen, the control policy characteristics are the same as in the previous case, but now all the three axes are controlled for all the cost functions. However, it can be noted that for the minimum-time case, the control is mainly bang-bang, as expected. For the minimum-effort maneuver, the control is mainly bang-off-bang, with a small non-zero control interval around the middle of the maneuver introduced to properly avoid the cones. Finally, for the minimum-energy case, the control policies for the three axes show a quasi-linear behavior. The angular velocity and the attitude history are consistent with the control inputs. The trajectories of the optical axis are reported in Fig. 6.7 for the three cost function. The motion has been projected onto a properly-defined latitude-longitude plane. The three trajectories are completely feasible as they do not enter the cones, they satisfy the boundary conditions and show little variations changing the cost function.

To verify the reliability of the results, the distributions of the results obtained through 1000 different simulations are reported in Fig. 6.8. As can be seen, for each of the three problems the optimizer converges to a single solution.

It should be noted that, using a pseudospectral optimizer [57], the obtained results are: 2.0346 τ_t for the minimum-time maneuver (the IPSO mean result is 2.1052 τ_t), 0.01635 $\tau_u^2\tau_t$ for the minimum-energy maneuver (the IPSO mean result is 0.01628 $\tau_u^2\tau_t$), 0.3648 $\tau_u\tau_t$ for the minimum-effort maneuver (the IPSO mean result is 0.3836 $\tau_u\tau_t$). The better results of the IPSO are obtained for the minimum-energy case where the expected optimal control is not discontinuous. The average number of iterations required by the IPSO is 2003 for the minimum-time case, 1420 for the minimum-energy case and 2180 for the minimum-effort case, leading to similar computational time for the three maneuvers (55.7 seconds for minimum-time, 53.1 for minimum-energy and 66.4 for minimum-effort maneuvers). The pseudospectral optimizer generally outperforms the IPSO in terms of cost function. However, especially with the minimum-effort maneuver, the pseudospectral optimizer must

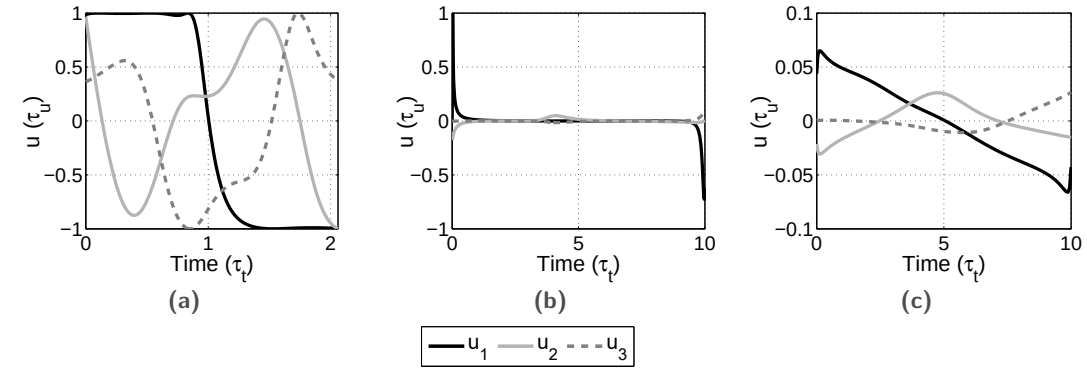


Fig. 6.4: Control policy for minimum-time (a), minimum-effort (b) and minimum-energy (c) maneuvers with cone constraint.

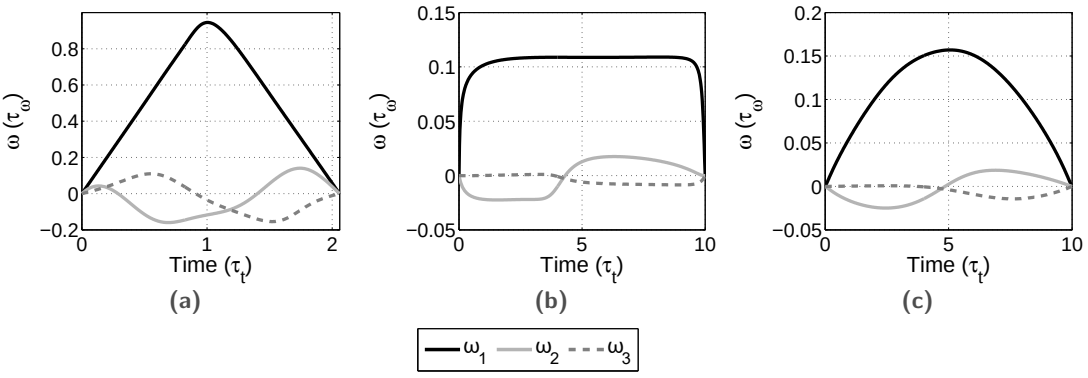


Fig. 6.5: Angular velocity histories for minimum-time (a), minimum-effort (b) and minimum-energy (c) maneuvers with cone constraint.

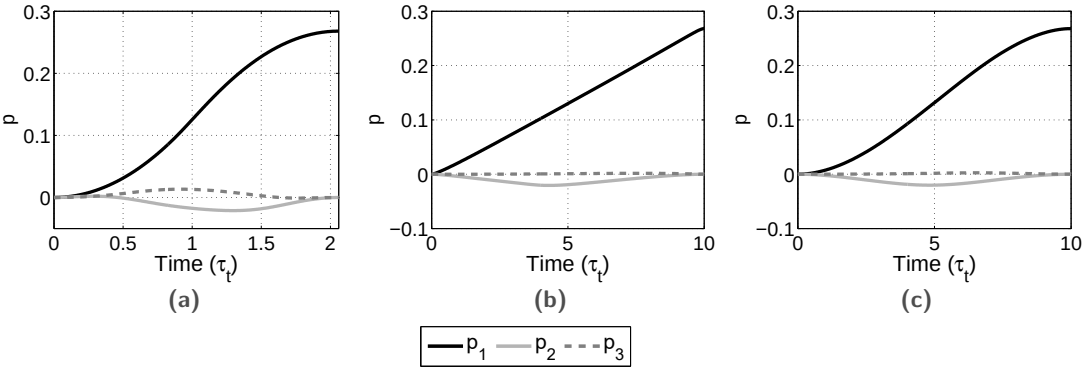


Fig. 6.6: Attitude histories for minimum-time (a), minimum-effort (b) and minimum-energy (c) maneuvers with cone constraint.

be used with high values of the mesh tolerance, otherwise the computational effort can be quite high[1]. The time required by the pseudospectral optimizer depends on the mesh tolerance and the solver. Using snopt and a mesh tolerance set to 1e-9, 8.9 seconds are needed for the minimum-time maneuver and 95.7 seconds for the minimum-energy maneuver, on average. With a mesh tolerance equal to 1e-5, 73.3

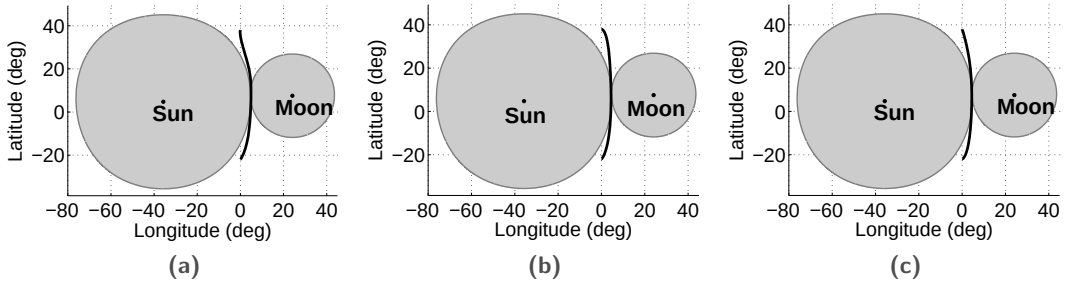


Fig. 6.7: Trajectories for minimum-time (a), minimum-effort (b) and minimum-energy (c) maneuvers with cone constraint.

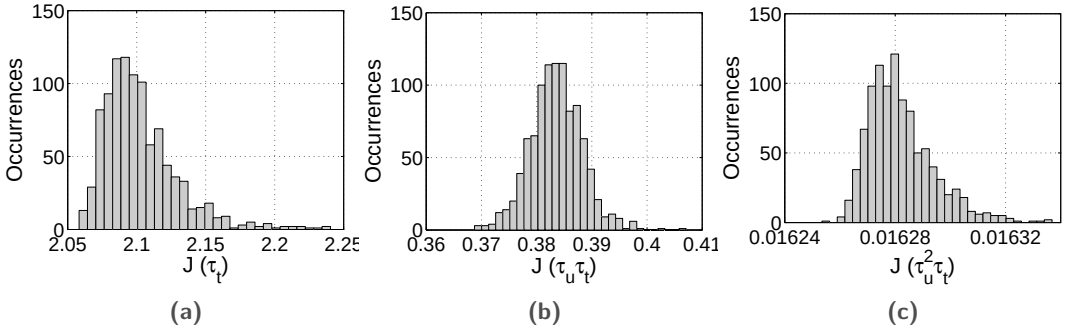


Fig. 6.8: Distribution of the IPSO results for minimum-time (a), minimum-effort (b) and minimum-energy (c) maneuvers with cone constraint.

seconds are required for the minimum-effort maneuver. It is noteworthy that the IPSO computational times are obtained in Matlab, and a C-conversion would reduce significantly the reported values. As a result, the IPSO has shown to be a reliable approach to get near-optimum solutions for the constrained BOCP examples.

6.5 Endnotes

In this chapter, Bolza optimal control problems have been addressed considering a constrained slew maneuver as a test case. Using the Inverse-dynamics Particle Swarm Optimization it has been shown that near optimal solutions can be found in the presence of nonconvexities concerning the cost function and the state constraints.

The major result of this chapter is that the common problem introduced by the differential flatness parametrization, that is the nonconvexity issues, is overcome by the global search ability of the particle swarm optimization. The combined usage of the differential flatness and the particle swarm algorithm allows a reduction in the number of unknown parameters and computational effort. No significant convergence differences have been noted among the solutions for minimum-time, minimum-energy and minimum-effort maneuvers. For minimum-effort and minimum-energy

problems, the computed maneuvers are quite similar to (and sometimes better than) the solutions given by a pseudospectral optimizer. For minimum-time maneuvers feasible near-optimal solution are found.

7

Particle Swarm Optimization with Domain Partition and Control Assignment for Minimum-Time Maneuvers*

Abstract

The Particle Swarm Optimization is employed to search for minimum-time maneuvers assigning a bang-bang control policy and dividing the search space into several sub-domains. Each sub-domain is defined by the number of switches per axis and the sign of the bangs. The algorithm searches for the minimum-time maneuver changing the time instants of the switches and moving from one sub-domain to another. The basic formulation of the Particle Swarm cannot guarantee a proper exploration of the search space but this issue is solved introducing the novel Push In and Push Out features. For general nonlinear problems, the maximum number of allowed switches is unknown and must be imposed based on some insight on the problem. Though the imposed bang-bang policy may lead to many local minima, the proposed approach can recognize the global minimizing solution when end-point conditions and path constraints are imposed. This formulation overcomes the limits of other techniques relying on swarm intelligence and mixed-integer programming where the control structure was obtained mapping intervals of real optimization variables to integer values. Two different test cases are reported to validate the method by comparison with other results from the literature.

Nomenclature

t	=	Time (s)	$(\cdot)^*$	=	Optimal quantity
$\boldsymbol{p}$	=	Modified Rodrigues Parameters	$\boldsymbol{\omega}$	=	Angular velocity
c	=	Generic constraint	$\tilde{(\cdot)}$	=	Penalty function weight
C	=	Generic penalty function	$\mathbf{x}$	=	Particle position
$\mathbf{v}$	=	Particle velocity	$\boldsymbol{X}$	=	Dimensionless state
$\boldsymbol{U}$	=	Dimensionless control	$N_{\mathcal{S}}$	=	Number of PSO particles
σ_i	=	Push-in/out parameters	M_S	=	Maximum number of switches
$(\cdot)_{0/f}$	=	Initial/final time value	$m_{S,j}$	=	Number of switches
J	=	Performance index	$N_{\mathcal{D}}$	=	Number of sub-domains

*This chapter is based on Ref. [124].

N_X	=	Number of state variables	N_U	=	Number of control variables
$\Delta_{(\cdot)}$	=	Constraint tolerance	R	=	Push-in/out parameters
$\mathcal{D}_I$	=	Integration domain	$s_{i,j}$	=	Switch time instants
δ_t	=	Time interval parameter	$r_{i,l}, r_{i,\overline{r}}$		random numbers for the i axis

7.1 Introduction

In Chapter 2, different approaches have been presented to formulate OCPs. Until now, most of the problems have been addressed using an inverse dynamics method based on a differential flatness formulation.

In Chapter 5, two opposite formulations have been outlined to numerically solve an optimal control problem. On one hand, one can impose the structure of the control policy and integrate the dynamics of the system to obtain the resulting kinematics. In this work, we refer to this procedure as the Direct Dynamics approach. On the other hand, if the dynamical system is differentially flat, the external control may be put as an explicit analytical function of some quantities, referred to as flat parameters, and their time derivatives. Inverse Dynamics approaches have been taken into account in the previous chapters. In this chapter, we choose to follow a Direct Dynamics approach imposing a bang-bang external control, where the switching time instants and the total maneuver time are the optimization variables.

Control-affine dynamical systems will be taken into account. For this class of problems, when the imposed constraints do not depend on the external control, PMP holds [29, 26]. Hence, if the dynamics does not show singular arcs, the control structure is purely bang-bang. However, for general non-linear systems, PMP can only establish that the control is bang-bang, but it cannot give the analytical solution of a minimum-time problem. In these cases, also the maximum number of switches is unknown. Therefore, the problem is to find the number of switches and the correct sign of the bangs. Boundary conditions and path constraints may be added to the optimization problem. Among all the feasible maneuvers (i.e. satisfying all the imposed constraints) we are interested in the minimum-time solution.

The proposed technique is based on PSO. A bang-bang control policy is imposed and the algorithm must select the proper number of switches for each axis and the sign of the bangs. The search space may be seen as the collection of several sub-domains, defined by the number of switches per axis and by the sign of the bangs. The particles may move among the sub-domains changing the values of the time instants related to the switches. Some of these sub-domains guarantee a feasible maneuver and other do not.

The method proposed in this chapter may lead to an optimization search space where many sub-optimal solutions can be detected. Accordingly, the local version of the PSO is employed to emphasize the exploration ability of the swarm which consists in comparing the different local minima in order to find the global minimum. However, the basic displacement model of the PSO is not sufficient to guarantee a proper exploration of the search space. Consequently, two new features have been introduced, namely the Push In and the Push Out displacements, helping the particle ability to change sub-domain affiliation and enhancing the possibility to try the maximum number of switches combinations. In this way, the PSO displacement is enforced and the risk to stop at local minimizing solutions is strongly reduced.

The numerical approach in Ref. [108] has inspired the present technique. However, the present formulation avoids the problems arising when reducing trajectory optimization problems to mixed-integer nonlinear programming. In fact, in [108] the control structure was obtained by mapping intervals of certain optimization real variables to integer values following a given set of rules. This desensitized the fitness function to even appreciable variations in the optimization variables. Consequently, a considerable fraction of the initial swarm population initially hovered around non-optimal regions of the search space, without moving the swarm toward better search regions and leading to long computational times. On the contrary, the proposed approach requires a less number of particles and converges in shorter computational times since there is no mapping between integer and real variables. Moreover, even very little variations of the optimization variables directly influence the integrated dynamics so we do not have the previously mentioned desensitization issue.

The algorithm requires to fix the maximum number of switches for each controlled axis. A good knowledge of the problem is required to set a proper value for this parameter. However, a solution strategy is proposed to tackle general optimization problems. Moreover, in [125] it is reported that for minimum-time point-to-point transfer of a general control-affine nonlinear system of dimension n, it is unlikely that a solution with more than $n-1$ switches satisfies Pontryagin's minimum principle. However, the same work reports that there may be exceptions.

Two different test cases will be presented. The first one concerns a minimum-time maneuver of a robotic arm: this problem is taken from [108, 126], where two different approaches for the solution have been presented. The second test case concerns the constrained slew maneuver already taken into account in Chapters 5 and 6. In this case, a detailed study of the characteristics of the search space is reported to emphasize the difficulty of the problem and to characterize the behavior of the proposed optimizer.

This chapter is organized as follows. Section 7.2 presents the hypothesis behind the assumption of the bang-bang control structure. Section 7.3 describes the implementation of the particle swarm and the control policy assignment. The partition of the search space is described in Sec. 7.4 while the constraint handling technique is outlined in Sec. 7.5. Section 7.6 presents two different test cases for the validation of the described technique. Finally, concluding remarks are given in Section 7.7.

7.2 Features of minimum-time maneuvers

We have seen, in Sec. 1.4.2, that for a control-affine dynamical system the minimum-time planning leads to a bang-bang control policy. For the sake of clarity, a summary of those results are here reported. This chapter deals with control-affine dynamical systems described by

$$\dot{\boldsymbol{X}}(t) = \mathbf{f}\left(\boldsymbol{X}\left(t\right)\right) + \boldsymbol{G}\left(\boldsymbol{X}\left(t\right)\right)\boldsymbol{U}\left(t\right) , \tag{7.1}$$

where $\boldsymbol{X}(t) : t \to \mathbb{R}^{N_X}$ is the continuous and piecewise differentiable state function and $\boldsymbol{U}(t) : t \to \mathbb{R}^{N_U}$ is the piecewise continuous external control function (refer to [29] for definitions about admissible state and control). N_X and N_U are the state and the control dimensions, respectively. The control $\boldsymbol{U}(t)$ is supposed to lie in an admissible region $[\boldsymbol{U}_{min}, \boldsymbol{U}_{max}]$. Eq. (7.1) may be nonlinear in the state (through the term $\boldsymbol{F}(\boldsymbol{X}(t))$) but is affine in the control. The operator $\boldsymbol{G}(\boldsymbol{X}(t))$ may be non-linearly dependent on the state and it is expressed by a $N_X \times N_U$ matrix. Equality boundary conditions and inequality path constraints are imposed as

$$\mathbf{b}(\boldsymbol{X}_0, \dot{\boldsymbol{X}}_0, \boldsymbol{X}_f, \dot{\boldsymbol{X}}_f, t_0, t_f) = \mathbf{0} \tag{7.2}$$

$$\mathbf{p}(\boldsymbol{X}(t), \dot{\boldsymbol{X}}(t), t) \leq \mathbf{0} \quad \forall t \in [t_0, t_f]. \tag{7.3}$$

Minimum-time maneuvers are considered, for which the performance measure to be minimized is

$$J = t_f \, . \tag{7.4}$$

Mathematical analysis of this class of problems can be found in standard texts on optimal control theory, for example in [26]. PMP [29] assures that the control must be extremal, i.e. $U = U_{min}$ or $U = U_{max}$ $\forall t \in [t_0, t_f]$ (assuming singular arcs do not exist). However, the number and the position of the switches are unknown as well as the sign of the bangs. The particle swarm optimization is employed to determine the optimal sequence of switches of the bang-bang control policy.

Accordingly, the local version of the PSO is employed to find the optimal sequence of switches for two different minimum-time optimal control problems.

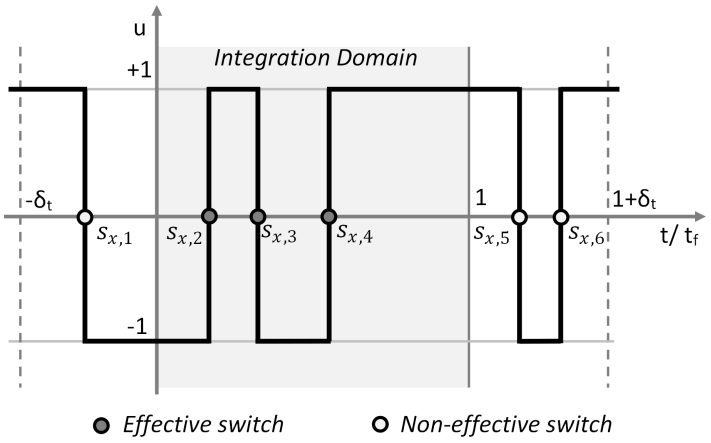


Fig. 7.1: Structure of the control policy assigned to the PSO particle.

7.3 Control structure assignment and search space partition

The most important input of the proposed algorithm in the maximum allowed number of switches, M_S. In terms of the decision variables, a particle's position can be expressed as

$$\mathbf{x} = [s_{x,1}, ..., s_{x,M_S}, s_{y,1}, ..., s_{y,M_S}, s_{z,1}, ..., s_{z,M_S}, t_f] \in \mathbb{R}^{3M_S+1}, \tag{7.5}$$

where $s_{i,j}$ is the j^{th} switch along the axis i ($j = 1, ..., M_S$, $i = x, y, z$) and t_f is the maneuver time. The time instant associated to the switch is evaluated as $s_{i,j}t_f$. An example control policy is reported in Fig. 7.1 for the x axis with normalized time and control and $M_S = 6$. The first bang on the left is imposed to be positive, which means that the overall sequence of bangs of each particle is the vector $[+1, -1, ..., (-1)^{M_S}] \in \mathbb{R}^{M_S+1}$, where the first value is always $+1$. For instance, in Fig. 7.1 the sequence of bangs is $[+1, -1, +1, -1, +1, -1, +1]$. Let $\mathcal{D}_I = [0, 1]$ be the (normalized) closed time interval of integration and $\mathcal{D}_E = (-\delta_t, 0) \cup (0, 1) \cup (1, 1+\delta_t)$ be the *extended* time interval, where δ_t is a user-defined parameter. By definition, $s_{i,j} \in D_E \ \forall j = 1, ..., M_S$, $i = x, y, z$. Since only the interval $\mathcal{D}_I$ is considered for the integration, the switches in $(0, 1)$ are called *effective* and the switches in $\mathcal{D}_E - \mathcal{D}_I$ are called *non-effective*. In Fig. 7.1, $\{s_{x,1}, s_{x,5}, s_{x,6}\}$ are non-effective switches whereas $\{s_{x,2}, s_{x,3}, s_{x,4}\}$ are effective switches. Non-effective switches may affect the integrated dynamics: in fact, having $s_{x,1} \in (-\delta_t, 0)$ and $s_{x,2} \in (0, 1)$ as in Fig. 7.1 makes the first *integrated* bang negative and the sequence of *effective* bangs is $[-1, +1, -1, +1]$. The extended time interval $\mathcal{D}_E$ allows the switches to enter and exit the integration domain $\mathcal{D}_I$ changing the control structure of each particle. Let $m_{S,i} \in \mathbb{Z}$, $i = x, y, z$, be the number of effective switches defined as:

- $m_{S,i} > 0$ means that $|m_{S,i}|$ effective switches are considered and the first bang is positive.

- $m_{S,i} < 0$ means that $|m_{S,i}|$ effective switches are considered and the first bang is negative.

With this new definition, $|m_{S,i}| \leq M_S$ for $i = x, y, z$. Once fixed $|m_{S,x}|$, $|m_{S,y}|$ and $|m_{S,z}|$, eight different combination of signs may be introduced. Accordingly, imposing that at least one switch per axis must be considered and that the bang in $(-\delta_t, s_{i,1})$ is fixed to +1, the number of sub-domains is given by $N_{\mathcal{D}} = (M_S - 1)^3$. Note that bangs shorter than the user-defined threshold $\Delta\tau_{min}$ are not taken into account by the optimizer.

7.4 Migration among sub-domains

Every combination of effective switches for the three axes, together with the sign of the first bang for each axis, determines a sub-domain where the search for the optimal maneuver may be accomplished. Let $\mathcal{C} = \{d_l,\, l = 1, ..., N_{\mathcal{D}} \,|\, d_l = [m_{S,x}^{(l)}, m_{S,y}^{(l)}, m_{S,z}^{(l)}],\, m_{S,i}^{(l)} \in \mathbb{Z},\, 1 \leq |m_{S,i}^{(l)}| \leq M_S,\, i = x, y, z\}$ be the collection of all the sub-domains d_l. The algorithm performs two fundamental tasks:

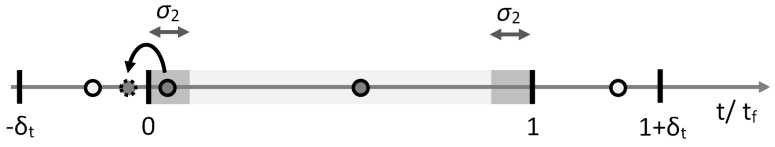
1. Understand if a sub-domain d_l is *feasible*, i.e. if the combination of effective switches $[m_{S,x}^{(l)}, m_{S,y}^{(l)}, m_{S,z}^{(l)}]$ allows to satisfies the end-point conditions and the path constraints.

2. If d_l is *feasible*, optimize the values of $s_{i,j}$ and minimize t_f to find the time-optimal maneuver.

To perform these two tasks, the particles must have the possibility to change sub-domain in $\mathcal{C}$ by changing the number of switches $s_{i,j}$ inside and outside the integrated time domain $\mathcal{D}_I$. Accordingly, the ability to thoroughly explore all the sub-domains must be strengthened to the maximum possible extent. This point justifies the implementation of the local version of the PSO. However, the velocities of the local PSO may not be sufficient to make the swarm explore all the feasible sub-domains since movements of non-effective switches outside $\mathcal{D}_I$ do not affect the integrated dynamics. With this regard, a slight modification to the original PSO has been implemented. First, let ρ, $r_{i,l}$ and $r_{i,r}$ be uniformly distributed numbers in $[0, 1]$ with $i = x, y, z$ and σ_1, σ_2 be two user-defined constant parameters. Second, a variable parameter $R^{(k)}$ is modeled as

$$R^{(k)} = R_0 - (R_0 - R_f) \min\left(1, \frac{k-1}{K}\right) \tag{7.6}$$

(a) Push-in



(b) Push-out

Fig. 7.2: Graphical representation of the push-in and push-out features.

After the application of the standard PSO displacement described in Eq. (3.1), two new operations are applied, the "push-in" and the "push-out" displacement, defined as follows:

Push-in: With reference to Fig. 7.2a, non-effective switches in $(-\delta_t, 0)$ and $(1, 1+\delta_t)$ are forced to move inside the integration domain $\mathcal{D}_I$ according to the following laws:

$$\forall s_{i,j}^{(k+1)} \in (-\delta_t, 0)\,, \quad s_{i,j}^{(k+1)} = \begin{cases} s_{i,j}^{(k+1)} & \text{if } r_{i,l} \leq 1 - R^{(k)}\,, \\ s_{i,j}^{(k+1)} + \sigma_1\rho & \text{if } r_{i,l} > 1 - R^{(k)}\,, \end{cases} \tag{7.7}$$

$$\forall s_{i,j}^{(k+1)} \in (1, 1+\delta_t)\,, \quad s_{i,j}^{(k+1)} = \begin{cases} s_{i,j}^{(k+1)} & \text{if } r_{i,r} \leq 1 - R^{(k)}\,, \\ s_{i,j}^{(k+1)} - \sigma_1\rho & \text{if } r_{i,r} > 1 - R^{(k)}\,. \end{cases} \tag{7.8}$$

Push-out: With reference to Fig. 7.2b, effective switches close to the boundary of $\mathcal{D}_I$ are forced to move outside the integration time domain. Once defined

$$\underline{s}_{i,j} = \min_{j|s_{i,j}>0}(s_{i,j})\,, \quad \bar{s}_{i,j} = \max_{j|s_{i,j}<1}(s_{i,j})\,, \tag{7.9}$$

the push-out displacement occurs according to the Eq. (7.10) and Eq. (7.11).

$$\text{If } \underline{s}_{i,j}^{(k+1)} < \sigma_2\,, \quad \underline{s}_{i,j}^{(k+1)} = \begin{cases} \underline{s}_{i,j}^{(k+1)} & \text{if } r_{i,l} \leq R^{(k)}\,, \\ -\underline{s}_{i,j}^{(k+1)} & \text{if } r_{i,l} > R^{(k)}\,. \end{cases} \tag{7.10}$$

$$\text{If } \bar{s}_{i,j}^{(k+1)} > 1 - \sigma_2\,, \quad \bar{s}_{i,j}^{(k+1)} = \begin{cases} \bar{s}_{i,j}^{(k+1)} & \text{if } r_{i,r} \leq R^{(k)}\,, \\ 2 - \bar{s}_{i,j}^{(k+1)} & \text{if } r_{i,r} > R^{(k)}\,. \end{cases} \tag{7.11}$$

Another numerical approach for solving minimum-time problem with a pre-assigned bang-bang control policy is reported in [108]. In that work, PSO has been proposed with a mixed-integer approach. The number of switches $m_{S,i}$ and the sign of the first bang $U_i(0)$ were integer values in $\mathbb{Z}$ indirectly related to the variation of some decision variables in $\mathbb{R}$. Supposing $\phi \in \mathbb{R}$ is the decision variable for $m_{S,i} \in \{1, 2, ..., M_S\}$, it was set $m_{S,i} = j$ if $\phi \in \Phi_j$ for $j \in \{1, 2, ..., M_S\}$, where $\Phi_j = [a_j, b_j)$ are non-intersecting sets with $b_j = a_{j+1}$. Similarly, setting ψ as the decision variable for $U_i(0)$ and $U_i(0) \in \{-1, +1\}$, $U_i(0) = -1$ if $\psi \in \Psi_1 = [-1, 0)$ and $U_i(0) = +1$ if $\psi \in \Psi_2 = [0, 1)$. As already pointed out by the authors in [108], the main drawback of this approach is that variations of ϕ and ψ only lead to variations of $m_{S,i}$ and $U_i(0)$ when they pass the border of one set going to another set. In this way, even large variations of ϕ and ψ may not lead to any variation of $m_{S,i}$ and $U_i(0)$. Moreover, when $U_i(0)$ switches from -1 to +1, the dynamics related to the particle drastically changes, creating a great amount of disorder in the swarm and slowing down the convergence. Differently from [108], the approach proposed in this work makes the variation of $m_{S,i}$ and $U_i(0)$ depend on the variations of the switch instants $s_{i,j}$. In this way, every small variation of the decision variables makes the dynamics change. Accordingly, smooth variations of the integration results are obtained helping the swarm to converge. The push-in and the push-out features give the swarm an enhanced ability to explore the SS but the values of σ_1 and σ_2 must be *small enough* so that the particles are only slightly perturbed.

7.5 Constraint handling

Solving minimum-time problems, the performance index associated with the optimization problem is to be chosen as the maneuver time. The performance index is selected in the form of an *Exterior Penalty Function* as described in Sec. 4.5.

First of all, it should be clear that the optimizer integrates the system dynamics which is consequently discretized over a finite set of points $\{t_k \in \mathbb{R} \,|\, t_0 < t_1 < ... < t_{N_T}\}$, where $N_T + 1$ is the number of discretization points. Accordingly, in a Direct Dynamics problem, initial conditions are automatically satisfied by the integrator. However, final conditions and path constraints must be considered while planning the time-optimal maneuver.

An Optimal Control Problem may be affected by equality an inequality constraints. The constraint functions do ultimately depend on the particle $\mathbf{x}_j$ and the discretized time instant t_k. From the numerical point of view, it is easier to transform an equality constraint into an inequality constraint. Consequently, the generic equality constraint $c(\mathbf{x}_i, t_k) = 0$ is treated as $|c(\mathbf{x}_i, t_k)| < \Delta$, where Δ is a user-defined tolerance. In the same way, the generic inequality constraint $c(\mathbf{x}_i, t_k) - \alpha \leq 0$ is *relaxed* and treated as

$c(\mathbf{x}_i, t_k) - \alpha \leq \pm\Delta$. Hence, all the constraints will be described as $c(\mathbf{x}_i, t_k) - \alpha \leq \pm\Delta$, where α is zero for equality constraints. The penalty function C associated with the constraint c is thus given as:

$$C_i(\mathbf{x}_j, t_k) = \begin{cases} 0 & \text{if } c_i(\mathbf{x}_j, t_k) - \alpha_i \leq \pm\Delta_i \,, \\ \dfrac{|c_i(\mathbf{x}_j, t_k) - \alpha_i| - \Delta_i}{\max(\Delta_i, \varepsilon_C)} & \text{otherwise.} \end{cases} \tag{7.12}$$

Note the normalization factor of the error is Δ_i until it is not zero. In this case, the convergence tolerance ε_C is used instead. Supposing to have N_c constraints, the performance index is thus given by

$$J_j = t_{f,j} + \sum_{i=1}^{N_C} \sum_{k=0}^{N_T} \tilde{c}_i \nu_{i,k} C_i(\mathbf{x}_j, t_k) + \tilde{n}\, N_{viol,j} \,, \tag{7.13}$$

where $\tilde{c}_i$ and $\tilde{n}_i$ are user-defined constant weights and $N_{viol,j}$ is the number of violated constraints of the j^{th} particle. The term $\nu_{i,k}$ is $1 \,\forall\, k \in \{0, ..., N_T\}$ if c_i is a path constraint while $\nu_{i,k}$ is δ_{k,N_T} if c_i a final condition, where $\delta_{l,m}$ is the usual Kronecker delta. Note that the normalization of the penalty functions in Eq. (7.12) is introduced in order to make their order of magnitude quite constant during the optimization process. In this way, the summation of the penalty functions with t_f in Eq. (7.13) is more consistent, i.e. the relative weights of the different terms remain quite fixed.

Every constraint C_i is considered with its tolerance Δ_i to help the convergence of the swarm. In this chapter *adaptive decreasing tolerances* are considered and the strategy described in Sec. 4.5.2 is employed. Note that the exit criterion for the optimizer is that all the imposed tolerances for the final conditions are lower than ε_C. Only for the path constraints, $\Delta_{(\cdot)}^{(\bar{k}+1)}$ is set equal to zero when its value is smaller than $10\varepsilon_C$. In this way, path constraints are completely satisfied before the exit condition.

7.6 Test cases

Two examples to validate the proposed technique are reported. Comparisons with other results in literature are given in order to verify the exactness of the obtained solutions. All the simulations have been carried out on a personal computer with an Intel®processor CoreTM i7-2670QM CPU @2.20GHz and with 6.00 GB of RAM. A Runge-Kutta integration scheme of order 4-5 is employed for the numerical integration, with relative and absolute tolerance referred to as ε_r and ε_a, respectively. Note that the number of discretization points N_T is chosen by the variable step size integrator in order to satisfy the integration tolerances ε_r and ε_a. All the constant

Tab. 7.2: Optimizer constant parameters.

Parameter	Value	Parameter	Value
C_p	1.5	$\tilde{c}_i$	10^6
C_l	2	σ_1, σ_2	0.01
N_γ	1500	δ_t	0.1
$\tilde{n}$	10^4	ε_C	10^{-8}
L_R	2	K	1000
w_0	1.4	w_f	0.6
R_0	0.5	R_f	0.25
$\Delta\tau_{min}$	10^{-3}	$\gamma^{(1)}$	0.1
ε_r	10^{-13}	ε_a	10^{-15}

parameters of the proposed optimizer are reported in Table 7.2. The only parameters that will be different in the following examples are the number of particles, N_S, and the values of the initial tolerances for the penalty functions, Δ_i, since their values depend on the examined problem.

The local version of the PSO has been selected for this work. Based upon the author' experience, the local paradigm shows superior convergence characteristics vis-à-vis the global version for the kind of problems addressed in this Chapter.

7.6.1 Minimum-time control of a two-link robotic arm

This problem is taken from [108, 126]. A planar robotic arm with two rigid and uniform links of mass m and length l is considered. A tip mass M is disposed as shown in Fig. 7.3. Actuators located at the shoulder and elbow joints provide torques U_1 and U_2, respectively. Both the torquers are independently subject to a control saturation constraint, i.e. $|U_i| \leq 1$, $i = 1, 2$. Considering frictionless joints, the arm dynamics is given by

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \\ \dot{X}_4 \end{bmatrix} = \begin{bmatrix} F^{-1}\mu_4\mu_5 & F^{-1}\mu_2\mu_5 & 0 & 0 \\ F^{-1}\mu_3\mu_5 & F^{-1}\mu_4\mu_5 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_1^2 \\ X_2^2 \\ X_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} \mu_2 & -(\mu_2+\mu_4) & 0 & 0 \\ \mu_4 & -(\mu_3+\mu_4) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ 0 \\ 0 \end{bmatrix}, \tag{7.14}$$

where the dimensionless quantities are defined by $\mu = M/m$, $\mu_1 = \mu + 1/2$, $\mu_2 = \mu + 1/3$, $\mu_3 = \mu + 4/3$, $\mu_4 = \mu \cos x_3$, $\mu_5 = \mu_1 \sin x_3$ and $F = 7/36 + 2\mu/3 + \mu_5^2$. As in [108, 126], we consider $\mu = 1$. As it can be seen, Eq. (7.14) is in the form of an control-affine system as in Eq. (7.1), with $N_X = 4$ and $N_U = 2$. Consequently, a bang-bang control can be adopted for minimum-time maneuvers.

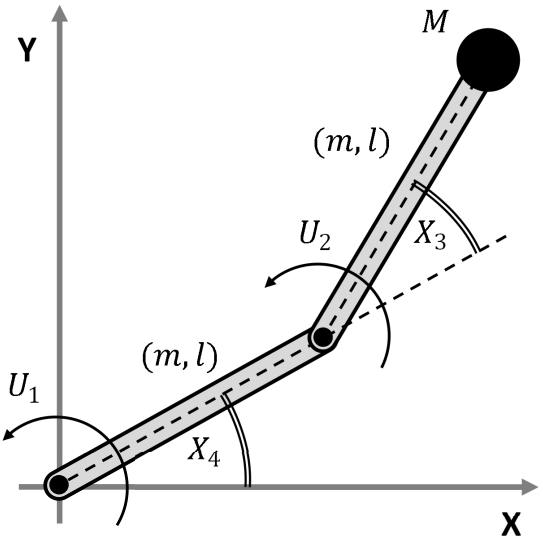


Fig. 7.3: Two-link robotic arm.

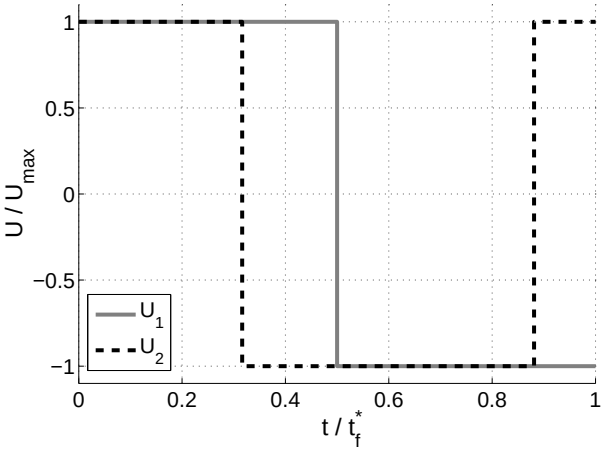


Fig. 7.4: Optimal control for the robotic arm.

Once the control is normalized, the independent variable time t is measured in units of $\kappa_t = 1/\sqrt{U_{max}/ml}$. Initial and final conditions of the problem are:

$$\boldsymbol{X}(t_0) = [0,\ 0,\ 0.5 \text{ rad},\ 0]^T, \quad \boldsymbol{X}(t_f) = [0,\ 0,\ 0.5 \text{ rad},\ 0.522 \text{ rad}]^T. \tag{7.15}$$

The performance index is chosen in the form of Eq. (7.13). No path constraints are taken into account and four penalty functions are requested for the four equality constraints concerning the final conditions in Eq. (7.15). The initial values of Δ_i related to $X_i(t_f)$, $i = 1, ..., 4$, have been set equal to 0.3. Results are reported for $M_S = \{2, 3, 4\}$; the number of particles $N_{\mathcal{S}}$ is 40 for $M_S = 2$, 60 for $M_S = 3$ and 80 for $M_S = 4$. Increasing the maximum number of switches more particles are required

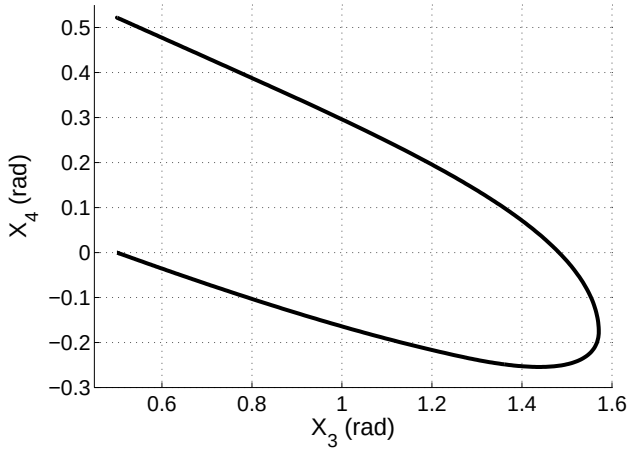


Fig. 7.5: Optimal state space trajectory for the robotic arm.

for the obtainment of the right result since more local minima are distributed in the search space. Preliminary tests have been carried out to choose appropriate values for $\mathscr{N}_{\mathcal{S}}$. Swarms with a smaller number of particles do not always guarantee the convergence toward the optimal solution. A discussion about how to chose proper values for M_S is given in Sec. 7.6.3.

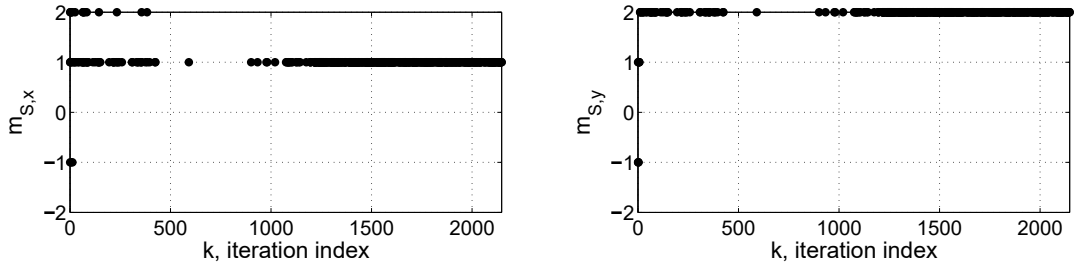
For all the studied cases, the algorithm is able to converge toward the optimal solution, which is given by $m_{S,x} = 1$ and $m_{S,y} = 2$. The optimal control in reported in Fig. 7.4, while the optimal state space trajectory for X_3 and X_4 is shown in Fig. 7.5. These results are perfectly consistent with those reported in [108, 126]: the switch for U_1 happens at $t/t_f = 0.5$ while the two switches for U_2 are located at $t/t_f^* = 0.3162$ and $t/t_f^* = 0.8811$. The maneuver time is equal to 2.9823 κ_t.

Note that, when trying to find a feasible maneuver imposing $M_S = 1$, so solution can be found. It means that, for this problem, we have found that the minimum-time maneuver corresponds to a maneuver given by the minimum *feasible* value of M_S.

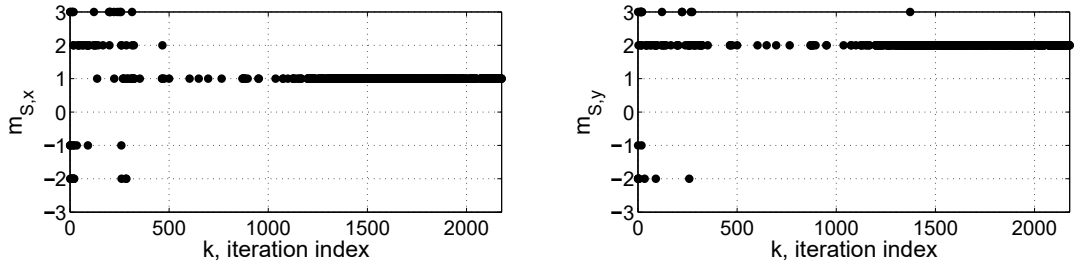
It is worth to see from Table 7.3 that the computational time required by the proposed algorithm is about 77 seconds when $M_S = 2$, 124 seconds for $M_S = 3$ and 190 seconds for $M_S = 4$. These results are evaluated over 100 different runs of the optimizer.

Tab. 7.3: Computational effort for the robotic arm problem.

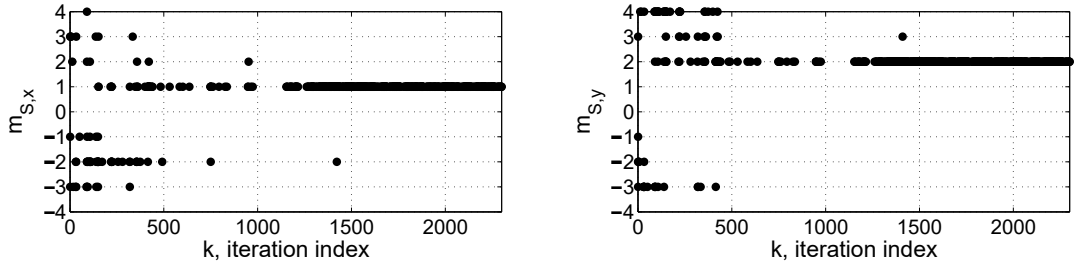
	$M_S = 2$	$M_S = 3$	$M_S = 4$
Number of particles, $N_{\mathcal{S}}$	40	60	80
Mean Computational Time (s)	79.81	125.83	190.33
Mean Number of Iterations	2196.10	2284.22	2398.93



(a) $M_S = 2$



(b) $M_S = 3$



(c) $M_S = 4$

Fig. 7.6: Convergence toward the optimal number of switches for different values of M_S.

This problem has been already solved in [108] with the technique that has been rapidly described toward the end of Sec. 7.4. In that case, 500 particles where needed. Even though several factors may impact on the computational time (processor, operative system, programming language and/or environment), the computational time required in [108] was on the order of several hours. The different number of required particles and the remarks made in Sec. 7.4 justify the difference in the computational times. Moreover, the proposed technique converges to a solution which perfectly coincides with the one reported in [108] obtained with SNOPT associated with a pseudo-spectral optimization software.

To conclude, it is quite interesting to see how the proposed technique is able to explore the search space and to converge toward the optimal solution. Even though Fig. 7.6 refers to three single cases with $M_S = 2$ in (a), $M_S = 3$ in (b) and $M_S = 4$ in (c), some general characteristics of the optimizer behavior may be inferred. The values of $m_{S,x}$ and $m_{S,y}$ corresponding to the temporary best solution during the

optimization process are reported and each point corresponds to an update of the best particle in the swarm. It can be seen that, for every value of M_S, the swarm is able to compare different combinations of switches and, after the search space has been properly explored, the swarm always recognizes the best maneuver and converges toward it. With the optimizer parameters reported in Table 7.2 and consistently with Table 7.3, the number of iterations required for the convergence is not strongly affected by M_S.

7.6.2 Minimum-time constrained slew maneuver

This problem has been already introduced in Sec. 5.2. In this case, the state is $\boldsymbol{X} = [\boldsymbol{p}^T, \boldsymbol{\omega}^T]^T$. Also in this case, Eq. (5.28) and Eq. (5.37) describe a control-affine dynamical system, , with $N_X = 6$ and $N_U = 3$. After the proper normalization, $|U_i| \leq 1,\ i = 1, 2, 3$ as in the previous example. Initial and final conditions are:

$$\boldsymbol{X}(t_0) = [0,\ 0,\ 0,\ 0,\ 0,\ 0]^T, \quad \boldsymbol{X}(t_f) = [0.2679,\ 0,\ 0,\ 0,\ 0,\ 0]^T. \tag{7.16}$$

The body-fixed reference frame $\boldsymbol{\mathcal{B}} = \{\hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2, \hat{\boldsymbol{e}}_3\}$ at time $t = t_0$ is taken as inertial reference frame and it will be referred to as $\boldsymbol{\mathcal{B}}_0 = \{\hat{\boldsymbol{e}}_{0,1}, \hat{\boldsymbol{e}}_{0,2}, \hat{\boldsymbol{e}}_{0,3}\}$.

$\boldsymbol{\omega} = \omega_1 \hat{\boldsymbol{e}}_1 + \omega_2 \hat{\boldsymbol{e}}_2 + \omega_3 \hat{\boldsymbol{e}}_3$

$\boldsymbol{U} = U_1 \hat{\boldsymbol{e}}_1 + U_2 \hat{\boldsymbol{e}}_2 + U_3 \hat{\boldsymbol{e}}_3$

All the reported solution are completely feasible with regard to the keep-out cone constraint. In fact, tolerances for the path constraints are set to zero in the final iterations of the optimization algorithm.

Before presenting the results, an important analysis concerning the distribution of the local minima in the search space is in order. When searching for maneuvers with a bang-bang control, some combinations of switches are feasible and other are not, as already stated in Sec. 7.4. In this particular case, the number of feasible maneuvers is extremely high and many local minima may be detected. The proposed algorithm has been forced to work with a prescribed number of switches, and all the possible combinations of switches have been tried considering $M_S = 1, ..., 7$. In this search space, 269 local minima have been found. With regard to the optimal maneuver time, which is 2.0078 κ_t =222.4080 seconds, the relative errors of these local minima span from 0.2% to 30%. Some of these extremal solutions are reported in Fig. 7.7: the number of switches for the x-body axis is fixed to $m_{S,x} = 1$ and $m_{S,x} = 3$, whilst $m_{S,y}$ and $m_{S,z}$ are considered from -7 to +7. The optimal solution is given by $m_{S,x} = 1$, $m_{S,y} = -2$ and $m_{S,z} = 3$ and it is specified with a gray circle.

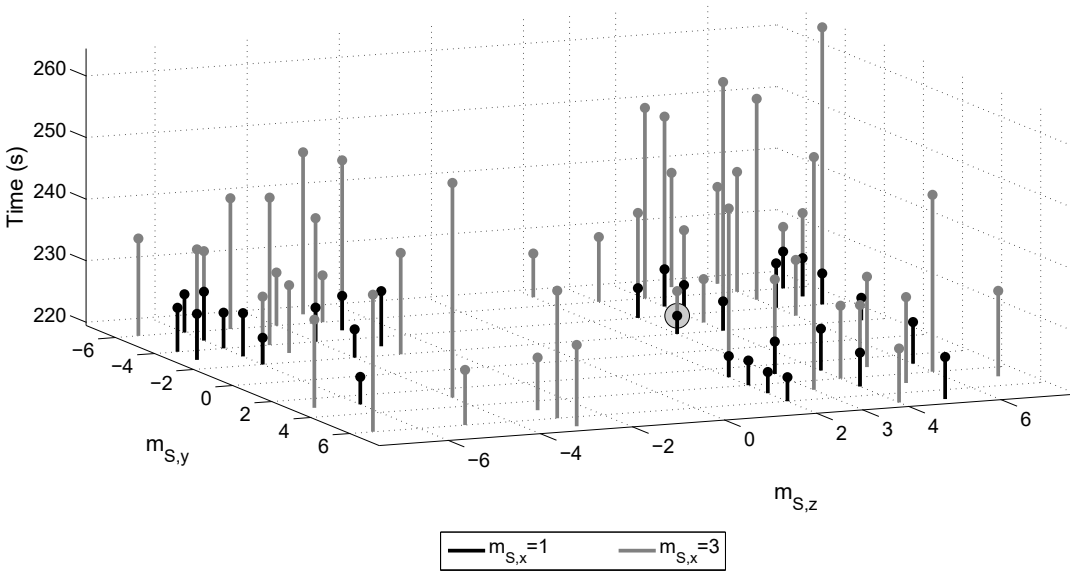


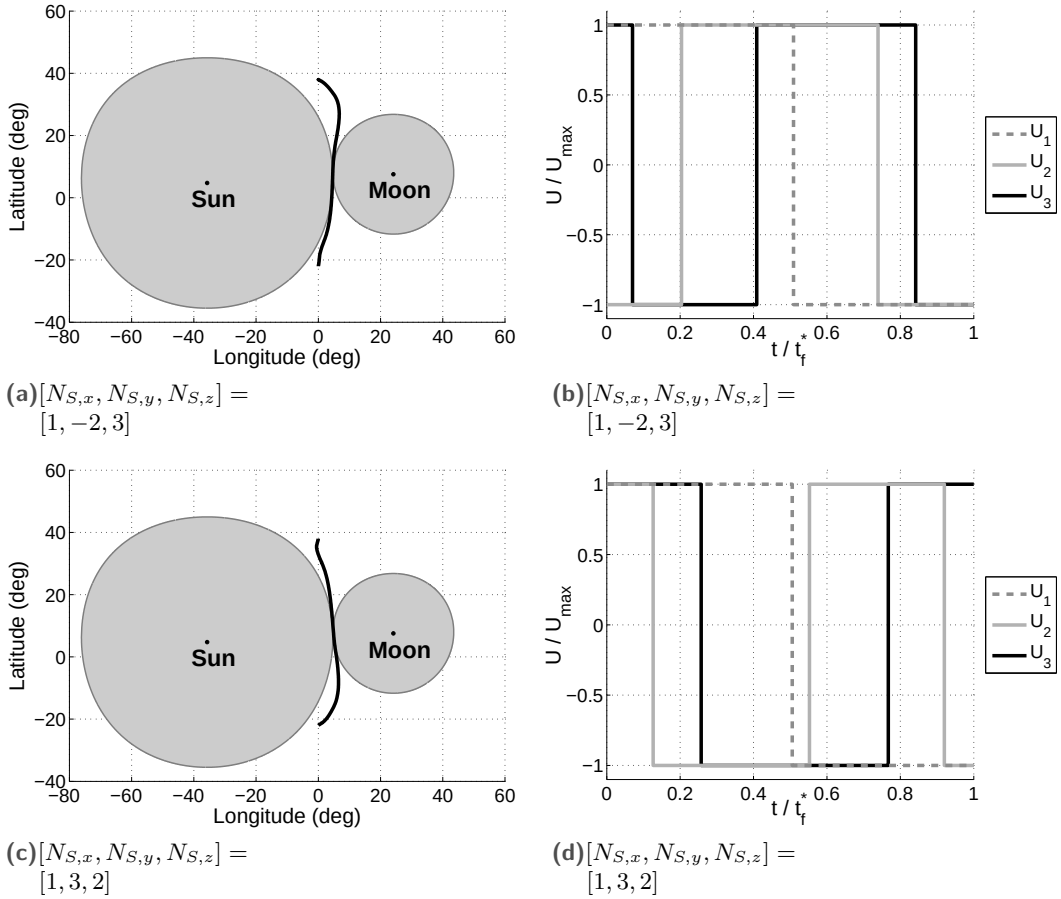
Fig. 7.7: Local minima in the search space of the slew maneuver problem.

The nearest local minimum is given by the combination $m_{S,x} = 1$, $m_{S,y} = 3$ and $m_{S,z} = 2$ and leads to a maneuver time of 2.0123 κ_t =222.9097 seconds. These two maneuvers are shown in Fig. 7.8. It can be seen that they take advantage of different nutational components of the motion (the first numerical evidence of time-optimal reorientation maneuvers with nutational components of the motion is given in [80]). In Fig. 7.8(a) the maneuver starts moving toward the right while in Fig. 7.8(c) the maneuver starts moving toward the left. In Fig. 7.8, the subplots (b) and (d) show the extremal control. By a visual inspection of the two trajectories, a slight symmetry of the the two maneuvers may be highlighted, which most probably is a consequence of the quasi-symmetrical geometrical disposition of the keep-out cones.

The time instants associated with the switches of the optimal and sub-optimal maneuver are reported in Table 7.4 as fractions of the optimal final time t_f^*.

Tab. 7.4: Switches obtained for the slew maneuver.

		Time instants of the switches (t_f^*)
$[m_{S,x}, m_{S,y}, m_{S,z}] = [1, -2, 3]$	along $\hat{\boldsymbol{e}}_1$	0.5089
	along $\hat{\boldsymbol{e}}_2$	$[0.2039\,,\ 0.7394]$
	along $\hat{\boldsymbol{e}}_3$	$[0.0703\,,\ 0.4089\,,\ 0.8419]$
$[m_{S,x}, m_{S,y}, m_{S,z}] = [1, 3, 2]$	along $\hat{\boldsymbol{e}}_1$	0.5063
	along $\hat{\boldsymbol{e}}_2$	$[0.1292\,,\ 0.5595\,,\ 0.9233]$
	along $\hat{\boldsymbol{e}}_3$	$[0.2590\,,\ 0.7689]$



(a) $[N_{S,x}, N_{S,y}, N_{S,z}] = [1, -2, 3]$

(b) $[N_{S,x}, N_{S,y}, N_{S,z}] = [1, -2, 3]$

(c) $[N_{S,x}, N_{S,y}, N_{S,z}] = [1, 3, 2]$

(d) $[N_{S,x}, N_{S,y}, N_{S,z}] = [1, 3, 2]$

Fig. 7.8: Optimal (a,b) and sub-optimal (c,d) maneuvers with keep-out cone constraints.

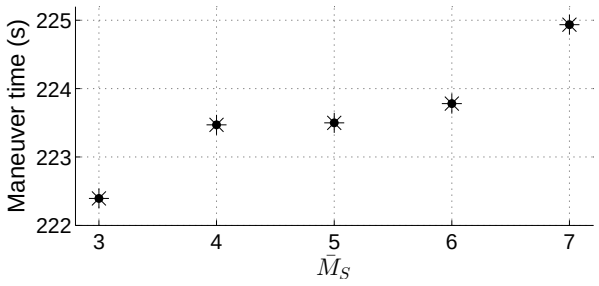


Fig. 7.9: Minimum time obtained when $\bar{M}_S$ switches occur on at least one of the three axes.

It is worth noting that, trying any combination of switches and imposing a maximum of 2 switches for all the body axes, no feasible maneuver has been found. Consequently, also in this problem, we can say that the time-optimal maneuver coincides with one of the maneuvers with the minimum *feasible* number of switches. Moreover, in this particular case, no feasible maneuver exists when only one switch is considered along $\hat{e}_2$ or when 2 switches are considered along $\hat{e}_2$ and less than 3 switches are considered along $\hat{e}_3$. If we define $\bar{M}_S$ as the number of switches to be guaranteed at least by one of the three axes, then we can see from Fig. 7.9 that the

maneuver time gets higher increasing the value of $\bar{M}_S$. This result justify the fact that it is reasonable to search for the optimal maneuver with low values of M_S.

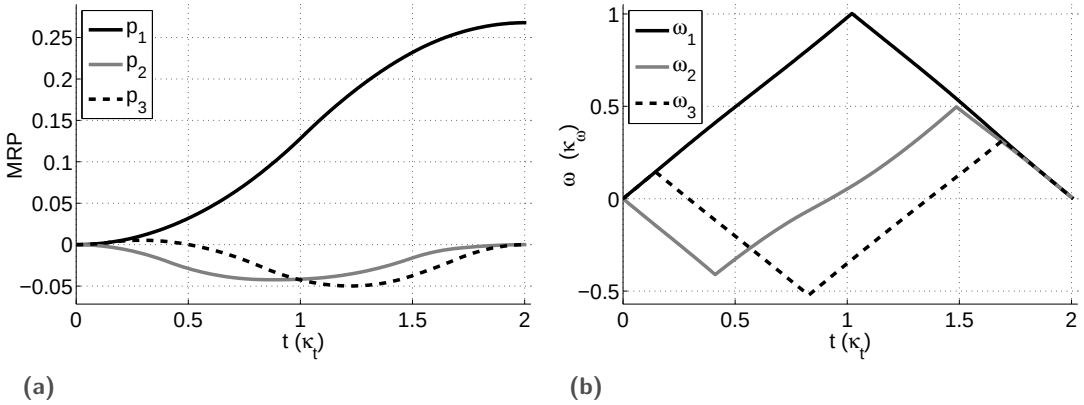
The kinematic profile in terms of Modified Rodrigues Parameters and angular velocities is provided for the optimal maneuver with $[m_{S,x}, m_{S,y}, m_{S,z}] = [1, -2, 3]$ in Fig. 7.10.

Until now, we have only tried all the possible combinations of switches to find out the minimum-time maneuver. This procedure requires a lot of time since the PSO algorithm is called for every combination. The proposed approach avoids this annoying time-consuming procedure. Setting $M_S = \{3, 4\}$, we have verified that the described optimizer is able to converge to the optimal solution. From previous numerical solutions and from the previous analysis we already know that $M_S = 3$ is the optimal value, but a generalization of the optimization procedure is considered in Sec. 7.6.3.

Setting $M_S = 3$ and $N_S = 70$ and having applied the proposed approach 100 times over the described test case, the following remarks are in order:

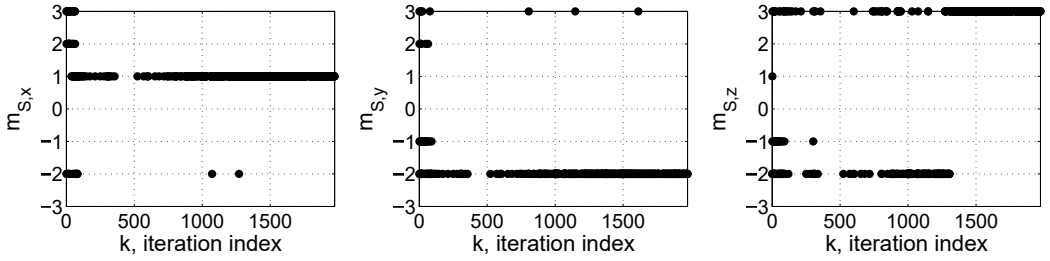
1. The optimal maneuver $[m_{S,x}, m_{S,y}, m_{S,z}] = [1, -2, 3]$ has been obtained for 71 times and the suboptimal maneuver $[m_{S,x}, m_{S,y}, m_{S,z}] = [1, 3, 2]$ for 29 times. However, given the fact that the error of the suboptimal maneuver is only of 0.2%, this result is related to the fact that the swarm cannot distinguish the optimal solution between the two options. Numerical experiments have shown that this characteristic behavior does not change increasing the number of particles.

2. The average computational time for convergence toward $[m_{S,x}, m_{S,y}, m_{S,z}] = [1, -2, 3]$ is 203.69 seconds with 2009.55 PSO iterations, whilst the average

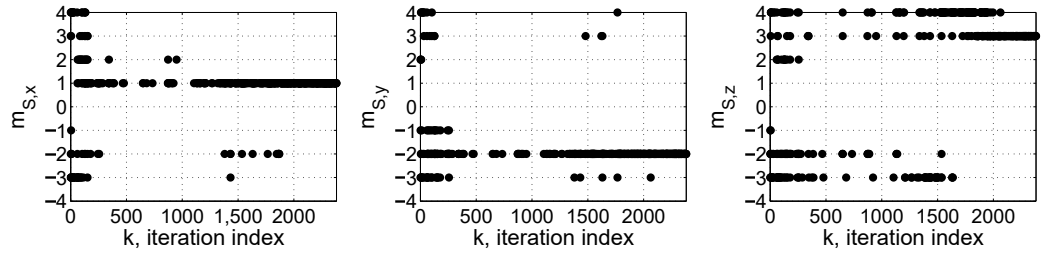


(a) (b)

Fig. 7.10: Optimal kinematics laws with keep-out cone constraints for $[\mathbf{m_{S,x}}, \mathbf{m_{S,y}}, \mathbf{m_{S,z}}] = [\mathbf{1}, -\mathbf{2}, \mathbf{3}]$.



(a) $M_S = 3$



(b) $M_S = 4$

Fig. 7.11: Convergence toward the optimal number of switches for different values of M_S.

computational time for convergence toward $[m_{S,x}, m_{S,y}, m_{S,z}] = [1, 3, 2]$ is 257.99 seconds with 2513.41 PSO iterations

3. The algorithm is still able to work with higher values of M_S; also in these cases the optimizer can give the optimal or the sub-optimal maneuver. As for the previous problem in Sec. 7.6.1, Fig. 7.11 reports the evolution of the optimal number of switches during the optimization process for two cases where the optimal sequence has been detected. It can be seen that, especially when $M_S = 4$, the algorithm spends a lot of time comparing different combinations of switches. With $M_S = 4$, 100 particles have been employed.

The errors on the final conditions for a generic solution are reported in Table 7.5. Note that the convergence threshold ε_C reported in Table 7.2 is satisfied for all the state variables.

Tab. 7.5: Final state values for a sample experiment of the slew maneuver.

State Variable	Units	Output value at $t = t_f$	Absolute error
p_1	-	0.2679	$0.4980 \cdot 10^{-8}$
p_2	-	0	$0.6977 \cdot 10^{-8}$
p_3	-	0	$0.6288 \cdot 10^{-8}$
ω_1	κ_ω	0	$0.1314 \cdot 10^{-8}$
ω_2	κ_ω	0	$0.6573 \cdot 10^{-8}$
ω_3	κ_ω	0	$0.3916 \cdot 10^{-8}$

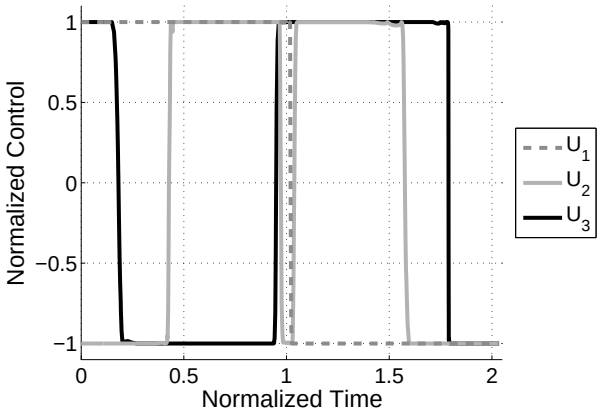


Fig. 7.12: Control policy given by the Pseudo-spectral optimizer.

With regard to the previous point 1., it is interesting to see that the presence of very close local minima is consistent with the analysis of re-orientation maneuvers without path constraints performed in [84]. In that case, reproducing the famous case study of [80], it was pointed out that local minima with different number of switches may exist leading to very little maneuver time differences (below 1% with respect to the optimal one).

Note that applying a pseudo-spectral approach with the software GPOPS-II (under the research activities license, [57]), the sub-optimal solution reported in Fig. 7.12 is obtained, where $[m_{S,x}, m_{S,y}, m_{S,z}] = [1, -4, 3]$. This maneuver is associated with a final time of 2.0346 κ_t = 225.3680 seconds (no appreciable variations have been appreciated changing the NLP solver or other setting parameters). Our numerical experiments have verified that the proposed approach and GPOPS-II give the same solution when no keep-out cone is considered, whilst our approach performs better when the keep-out cone is considered. A reasonable explanation lies in the different approximation method used for the control: GPOPS-II approximates the control (and the state) with Lagrange polynomials over the collocation points, while the proposed method impose the (*already known*) optimal bang-bang solution and properly integrate the dynamics.

7.6.3 Additional remarks

The solution of the two problems in Sec. 7.6.1 and Sec. 7.6.2 are extremely useful in order to suggest the criteria for using the proposed technique in general minimum-time problems. It has been underlined that an insight into the optimization problem is helpful to guess the minimum reasonable value of M_S. In the two problems, we based our guess on the knowledge of previous solutions from the literature. However, when solving new problems where it's not easy to guess the proper value of M_S,

the solving approach may be to start from low values of M_S. In Sec. 7.6.2 it has been underlined that using $M_S = 1$ and $M_S = 2$ no feasible solutions exist and the minimum-time maneuver is associated with $M_S = 3$. Moreover, solving the robotic arm problem with $M_S = 1$, no solution has been found, and also here the minimum-time maneuver has been found with the minimum feasible value of M_S which is 2. Even though there is no theoretical evidence that this is the general rule, these results suggest that the minimum-time solution is likely to be associated to low *feasible* values of M_S. If the optimization problem is convex with regard to M_S, than a good solving strategy is to start from the minimum value of M_S and let M_S increase until we get a decrease in the maneuver time.

7.7 Endnotes

In this chapter, a novel approach has been proposed for planning time-optimal maneuvers imposing a bang-bang external control. The optimizer is based on the Particle Swarm Optimization and only requires to set the maximum number of switches allowed for each axis. With the reported examples, it has been shown that several local minima may exist when using the proposed approach. However, thanks to the partition of the search space and the introduction of the Push In and the Push Out features, the optimization algorithm is able to recognize the global minimizing solution so that the convergence toward the bang-bang optimal maneuver described by the Pontryagin Maximum Principle is guaranteed.

Two different test cases have been analyzed and solved to validate the optimizer. In the first example, characterized by four state-space variables and no path constraints, the convergence toward the optimal solution has been demonstrated with different values of the maximum number of switches. For the second example, described by six state-space variables and non-linear path constraints, the optimal solution is reached in more than 70% of cases over 100 simulations, path constraints are completely satisfied and errors on the final conditions are lower than 10^{-8} in normalized units. Compared to other similar numerical approaches, the computational effort is rather small and, using a personal computer, a maximum time of few minutes are required for the convergence. Moreover, the second example has shown that the solution obtained with the described method can be better than the solutions obtained with pseudo-spectral optimization algorithms. The proposed optimizer represents a valuable tool when the optimal control is known to be a bang-bang policy.

Part IV

Planning of Spacecraft Formation Reconfigurations

8

Minimum-Time Reconfiguration Maneuvers of Satellite Formations Using Perturbation Forces*

Abstract

A novel approach for minimum-time reconfiguration of satellite formations is described considering the perturbation forces as control variables. Planning appropriate attitude maneuvers for each satellite, the atmospheric drag and of the solar radiation pressure are properly controlled and the formation is given the appropriate inputs to achieve the imposed reconfiguration. Limits and advantages of the presented maneuvers are examined considering Low Earth Orbits, Medium Earth Orbits and Geostationary Orbits. The Inverse Dynamics Particle Swarm Optimization is involved: the integration of the attitude dynamics is avoided, thus reducing the computational effort, and satisfied attitude constraints at the initial and final time instants are guaranteed. B-spline curves approximate the attitude kinematics, variable time mesh-points are introduced and adaptive decreasing tolerances are considered for the imposed constraints. The evolution of the configuration is simulated with a high-fidelity orbital simulator considering all the perturbations that can affect the maneuver. Two test cases are taken into account, one involving a circular formation reconfiguration and the other an along-track reconfiguration.

Nomenclature

$\boldsymbol{r}(t)$	=	Inertial position	$\boldsymbol{p}(t)$	=	External perturbation
μ	=	Earth gravitational constant	$\boldsymbol{\rho}$	=	Relative position
$\mathcal{I}$	=	ECI coordinate system	$\mathcal{L}$	=	LVLH coordinate system
t	=	Time	$\boldsymbol{\xi}$	=	Attitude parameters
$(\cdot)_{0/f}$	=	Initial/final time value	$(\cdot)^*$	=	Optimal value
$(\cdot)_{c/d}$	=	Chief/Deputy referred value	$\boldsymbol{xi}$	=	Attitude control torque
$\boldsymbol{\omega}$	=	Body angular velocity	$N_{\mathcal{F}}$	=	Number of formation spacecrafts
$\boldsymbol{X}$	=	Orbital states of the formation	$\boldsymbol{Y}$	=	Attitude states of the formation
$\boldsymbol{U}$	=	Attitude controls of the formation	$\boldsymbol{M}_{GG}$	=	Gravity gradient torque

*This chapter is based on Refs. [8, 127].

$\mathbf{æ}$	=	Classical orbital elements set	R, K_{zx}	=	Relative trajectory parameters
$c_{y/zx}$	=	Relative ellipse axes	φ_{zx}	=	Relative plane inclination
J_2	=	Gravitational parameter	t	=	Time
C_D	=	Drag coefficient	m	=	Satellite mass
ϱ	=	Atmospheric density	α, β	=	Orientation angles
$\hat{(\cdot)}$	=	Unit vector	γ	=	Reflectivity coefficient
$\mathcal{S}_0$	=	Solar radiation constant	γ	=	Reflectivity coefficient
$\zeta(t)$	=	Shadow function	$\nabla \boldsymbol{g}_{J_2}$	=	GG acceleration due to J_2
I	=	Inertia tensor	J	=	Performance index
$\mathcal{P}$	=	Penalty function	N_{viol}	=	Number of violated constraints
Δ	=	Decreasing tolerance	∂_{min}	=	Minimum inter-distance
$\tilde{(\cdot)}$	=	Approximation coefficient	N_T	=	Discretization points
$N_{\mathcal{S}}$	=	Number of PSO particles	$N_{\mathcal{P}}$	=	Number of B-spline parameters

8.1 Introduction

This chapter describes a spacecraft formation reconfiguration strategy based on the coupling between translational and rotational dynamics introduced by the perturbations. It is demonstrated that modifying the attitude of the satellites, the effects of the atmospheric drag and of the solar radiation pressure can change in such a way to give the formation the appropriate inputs to achieve the imposed reconfiguration. The mission scenario is inspired by that of the JC2Sat Formation Flying Mission [128].

Satellite formation flying (SFF) is an important research theme of the recent scientific literature in the astronautical field: missions based on SFF may reduce the overall costs and benefit from the distribution of the payload. Guidance, navigation and control of SFF play a fundamental role in guaranteeing the proper inter-linking and geometrical distribution of the satellites which are needed for the fulfillment of the mission [129, 130].

The study of SFF is easily modeled as a relative motion problem described by the linearized Hill-Clohessy-Whiltshire equations [131]. However, great effort has been spent in developing non-linear models of the relative motion; most of the recent results may be found in [132].

The planning of reconfiguration maneuvers is usually based on the following assumptions: 1) maneuvers are accomplished by means of an impulsive or continuous control based on fuel consumption; 2) rotational dynamics is not modeled. Analytical solution may be found only for simplified problems, as in [133], hence numerical techniques are usually employed to find optimal solutions: the Multiple-Shooting method is applied in [134], a Pseudospectral technique is involved in [135] whilst a Mixed-Integer Linear Programming is presented in [136]. Heuristic methods are

also usually applied: the Particle Swarm Optimization is used in [137], the Brain Storm Optimization is exploited in [138] and a Genetic Algorithm are adopted in [139].

Only few works take the attitude dynamics into account when reconfiguring a satellite formation. For instance, in [140] attitude maneuvers are required to properly orient the body-fixed thrusters, while in [141] several attitude constraints are imposed for deep space missions. However, none of these works consider the coupling between the translational and the rotational dynamics introduced by the perturbations.

In order to guide and control a SFF, small forces are generally required and perturbations may be used to accomplish the satellites operations. This possibility has been already investigated as far as the maintenance issue is concerned. The differential drag has been firstly taken into account for the formation-keeping in [142] with a simple feedback control law. A great number of works have than been produced considering the problem of the rendezvous between two or more satellites; some interesting results may be found in [143, 144] where the J_2 effect is taken into account and in [145, 146] where an adaptive Lyapunov control strategy is presented. For very low Earth orbits, also the lift force may be considered [147]. The problem of the formation maintenance using differential drag has been recently represented in [147, 148, 149]. Only in [150], however, the problem of the reconfiguration is taken into account. Also the solar radiation pressure has been considered for station-keeping problems [151] or formation mainteinance problems [152, 153]. Some reconfiguration maneuvers have been presented in [154, 155]. Finally, an interesting application of the geomagnetic Lorentz force for formation reconfiguration maneuvers has been reported in [156].

As previous works have already investigated the formation maintenance problem, this work does instead focus on the opportunity to perform reconfiguration maneuvers by means of perturbation forces. In particular, modifying the attitude of the satellites, the effects of the atmospheric drag and of the solar radiation pressure change; controlling these perturbation forces, the appropriate inputs are given to the maneuverable satellites to achieve the desired relative configuration. Limits and possibilities offered by using the drag perturbation and the solar radiation pressure perturbation (individually or combined, depending on the altitude) will be analyzed considering Low Earth Orbits (LEO), Medium Earth Orbits (MEO) and Geostationary Orbits (GEO). Minimum-time reconfiguration maneuvers are investigated: since perturbation forces have a small intensity compared to the gravitational field forces, the minimum-time planning guarantees to exploit the effect of the perturbation forces at the maximum extent and to obtain reasonable maneuver times. Several

constraints are imposed, such as the final configuration, the collision avoidance and the maximum available torque for the attitude maneuvers.

The IPSO is employed, as it is a fast and validated optimization method suitable for the proposed problem where the integration of the full non-linear orbital dynamics is already a heavy time-consuming process. In fact, reliable results may be obtained only considering both the SFF orbital dynamics (through an high fidelity orbital simulator) and the attitude dynamics in order to properly model the intensity and the direction of the perturbation forces during the maneuver. Differently from other commonly used numerical approaches, the IPSO requires a reduced number of optimization parameters thus making it possible to deal with very complex optimization problems. As proved in the previous chapters, the IPSO generally requires small computational efforts since the integration of the attitude dynamics is avoided. Furthermore, satisfied boundary conditions for the attitude are guaranteed, thus reducing the complexity of the performance index. B-spline curves are used to model the attitude kinematics of the satellites: an innovative way for modeling the attitude profiles of the satellites is introduced, considering control points with variable attitude parameters and time mesh-points. Moreover, a novel strategy for adaptive decreasing tolerances has been adopted.

The main objective of this chapter is to show the limits and possibilities offered by the proposed maneuver approach by means of:

1. Theoretical investigations: starting from simple mathematical models (Hill-Clohessy-Wiltshire (HCW) equations and Gauss' Variational Equations (GVE)), the relative motion variations induced by the drag and the solar radiation pressure will be investigated to understand how they affect the proposed trajectories.

2. Numerical simulations: accurate numerical experiments will prove the effectiveness of the theoretical insights.

It is noteworthy that in this chapter the IPSO method is utilized to solve a very hard problem which requires an efficient work-station in order to obtain the solution in a reasonable time, in contrast to the applications reported from Chapter 5 to Chapter 6, where the IPSO has been presented as an advantageous method for possible on-board implementation of an autonomous path planning. In fact, the goal of this chapter is to demonstrate the feasibility of the proposed maneuver approach, verifying the theoretical insights through a validated numerical optimization technique.

This chapter is organized as follows. Section 8.2 presents a description of the relative motion problem considering non-linearities, external control and orbital perturbation. Section 8.3 introduces some reference trajectories described by linear

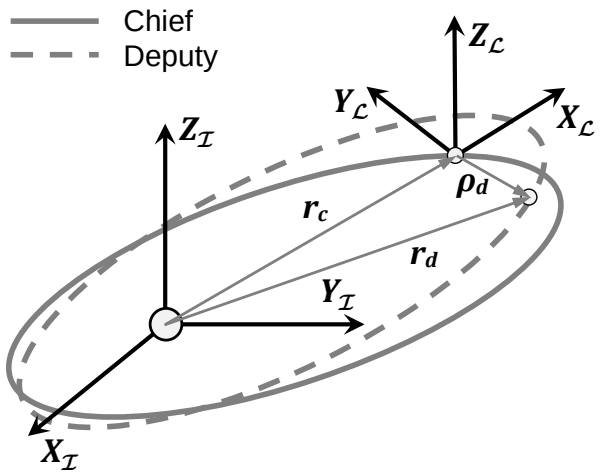


Fig. 8.1: ECI coordinate system $\mathscr{I}$ and LVLH coordinate system $\mathscr{L}$.

relative motion models. Section 8.4 describes the perturbation-based approach to accomplish formation reconfiguration maneuvers. Section 8.5 describes the implementation of the IPSO strategy applied to the presented problem. Section 8.6 presents the numerical results obtained with some interesting test cases. Finally, concluding remarks are given in Section 8.7.

8.2 Satellite formation reconfiguration

Let us suppose to have a SFF with $N_{\mathcal{F}}$ satellites identified with the subscript $i \in \{1, 2, ..., N_{\mathcal{F}}\}$. We introduce a Earth Centered Inertial (ECI) coordinate system $\mathcal{I}$ as reported in Fig. 8.1. Considering the Keplerian term and N_{pert} environmental perturbations $\boldsymbol{p}_{j,i}$ acting on the i^{th} satellite, the equation of motion in $\mathcal{I}$ is

$$\ddot{\boldsymbol{r}}_i = -\frac{\mu}{r_i^3}\boldsymbol{r}_i + \sum_{j=1}^{N_{pert}} \boldsymbol{p}_{j,i} \tag{8.1}$$

where $\mu = 3.986 \cdot 10^5$ km^3/s^2 is the Earth gravitational constant. The terms $\boldsymbol{p}_{j,i}$ in Eq. (8.1) will be described in details in Sec. 8.4: the number of perturbation terms N_{pert} depends on the accuracy associated to the numerical model. The relative motion dynamics is obtained describing the motion of the i^{th} *deputy* satellite with respect to the motion of the *chief* satellite, here denoted by the subscript c. Note that the chief satellite position may be empty, which means that there is actually no satellite in $\boldsymbol{r}_c$. In such case, the chief position may be identified as the barycenter of the formation. With reference to fig. 8.1, we define the relative position vector $\boldsymbol{\rho}_i = \boldsymbol{r}_i - \boldsymbol{r}_c = [x_i,\, y_i,\, z_i]^T$. The relative motion dynamics is described in the Local-

Vertical, Local-Horizontal (LVLH) coordinate system $\mathcal{L}$ centered in the chief satellite position identified by $\boldsymbol{r}_c$.

In this chapter, the non-linear dynamics of every satellite will be taken into account for the numerical integration. The non-linear equations of relative motion may be expressed in $\mathcal{L}$ [132], but this representation does not reduce the numerical complexity of the problem when considering a perturbed reference orbit: in fact both the relative states of the *deputies* and the inertial state of the *chief* must be integrated. Accordingly, the numerical integration is performed on the inertial states and the relative formulation is only used for the *a posteriori* description of the formation.

Let us now suppose that, at a defined initial time instant $t_0 = 0$, the formation is disposed such that it satisfy some geometrical constraints related to the position and velocity initial conditions and that we want to reconfigure this formation. Consequently, we must impose precise final conditions to be satisfied after an appropriate maneuver completed at the time instant $t = t_f$. The purpose of this work is to explore the possibility to accomplish a formation reconfiguration maneuver by means of the atmospheric drag perturbation $\boldsymbol{p}_D$ and the solar radiation pressure perturbation $\boldsymbol{p}_{SRP}$. Both these two forces vary as a function of the attitude, which may be represented by the vector $\boldsymbol{\xi}$ in $\mathbb{R}^3$ or $\mathbb{R}^4$ depending on the chosen representation [114]. Hence, the perturbation forces are the inputs leading to the reconfiguration, but the attitude maneuver are the tool through which the reconfiguration maneuver is made possible. Since perturbation forces have a small intensity compared to the gravitational field forces, minimum-time reconfiguration maneuvers are searched for to exploit the effect of the perturbation forces at the maximum extent and to obtain reasonable maneuver times.

The minimum-time optimization problem is summarized as follows: find the minimum maneuver time t_f^* such that, $\forall i = 1, 2, ..., N_{\mathcal{F}}$, an optimal attitude kinematic law $\boldsymbol{\xi}_i^*(t)$ and the associated attitude control torque $\boldsymbol{u}_i^*(t) = [u_{i,1}^*(t),\, u_{i,2}^*(t),\, u_{i,3}^*(t)]^T$ are found which properly set the intensity and/or the direction of the perturbation forces thus leading from the initial to the final formation configuration. The configuration constraints do actually impose constraints on the values of position and velocities of all the satellites in the formation, for $t = t_0$ and $t = t_f^*$.

First, the following quantities and relationships are defined:

- $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_{\mathcal{F}}}] \in \mathbb{R}^{6N_{\mathcal{F}}}$ where $\boldsymbol{x}_i = [\boldsymbol{r}_i, \dot{\boldsymbol{r}}_i], \boldsymbol{r}_i \in \mathbb{R}^3$ is the inertial positions and $\dot{\boldsymbol{r}}_i \in \mathbb{R}^3$ is the inertial velocity of each satellite, expressed in $\mathcal{I}$.

- $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{N_{\mathcal{F}}}] \in \mathbb{R}^{6N_{\mathcal{F}}}$ where $\boldsymbol{y}_i = [\boldsymbol{\xi}_i, \boldsymbol{\omega}_i], \boldsymbol{\xi}_i \in \mathbb{R}^3$ is the attitude parameterization (given by roll, pitch and yaw) and $\boldsymbol{\omega}_i \in \mathbb{R}^3$ the body angular velocity, expressed in $\mathcal{B}$.

- $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{N_\mathcal{F}}] \in \mathbb{R}^{3N_\mathcal{F}}$ where $\boldsymbol{u}_i \in \mathbb{R}^3$ is the vector of the external torque for controlling the attitude of each satellite.
- $\boldsymbol{b}$ defines the initial conditions such that $\boldsymbol{b} : \mathbb{R}^{6N_\mathcal{F}} \times \mathbb{R}^{6N_\mathcal{F}} \to \mathbb{R}^{N_b}$.
- $\boldsymbol{e}$ defines the final conditions such that $\boldsymbol{e} : \mathbb{R}^{6N_\mathcal{F}} \times \mathbb{R}^{6N_\mathcal{F}} \to \mathbb{R}^{N_e}$.
- $\boldsymbol{p}$ defines the path constraints such that $\boldsymbol{p} : \mathbb{R}^{6N_\mathcal{F}} \times \mathbb{R}^{6N_\mathcal{F}} \times \mathbb{R}^{3N_\mathcal{F}} \times \mathbb{R} \to \mathbb{R}^{N_p}$.

The values of $N_b, N_e, N_p \in \mathbb{R}$ depend on the problem.

Denoting the j^{th} constraint at the *beginning* of the maneuver as b_j, the j^{th} constraint at the *end* of the maneuver as e_j, the j^{th} *control* constraint as c_j and the j^{th} *path* constraint as p_j, the mathematical formulation of the optimization problem is expressed as:

$$
\begin{aligned}
&\text{Find } \boldsymbol{X}(t), \boldsymbol{Y}(t), \boldsymbol{U}(t), t_f \in \mathbb{R} \\
&\text{minimizing} \\
&J = t_f - t_0 \\
&\text{subject to, } \forall t \in [t_0, t_f] \\
\text{orbital dynamics:} \quad &\ddot{\boldsymbol{r}}_i = \sum_{j=1}^{n_g} \sum_{k=1}^{n_g} \boldsymbol{\phi}_{j,k}(\boldsymbol{r}_i) + \boldsymbol{p}_{\odot,\oplus}(\boldsymbol{r}_i) + \boldsymbol{p}_{\leftmoon,\oplus}(\boldsymbol{r}_i) \\
&\qquad + \boldsymbol{p}_D(\boldsymbol{r}_i, \boldsymbol{\xi}_i, t) + \boldsymbol{p}_{SRP}(\boldsymbol{r}_i, \boldsymbol{\xi}_i, t) \\
\text{attitude dynamics:} \quad &\dot{\boldsymbol{\xi}}_i = \Xi(\boldsymbol{\xi}_i)\, \boldsymbol{\omega}_i \\
&\dot{\boldsymbol{\omega}}_i = I^{-1}(\boldsymbol{u}_i + \boldsymbol{M}_{GG}(\boldsymbol{r}_i, \boldsymbol{\xi}_i) - \boldsymbol{\omega}_i \times I\boldsymbol{\omega}_i) \\
\text{initial conditions:} \quad &\boldsymbol{b}(\boldsymbol{X}(t_0), \boldsymbol{Y}(t_0)) = \boldsymbol{0} \\
\text{final conditions:} \quad &\boldsymbol{e}(\boldsymbol{X}(t_f), \boldsymbol{Y}(t_f)) = \boldsymbol{0} \\
\text{path constraints:} \quad &\boldsymbol{p}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{U}, t) \leq \boldsymbol{0}
\end{aligned}
\tag{8.2}
$$

Initial and final conditions determine the satellite configuration at the beginning and at the end of the maneuver. Inside the path constraints, the collision avoidance (satellites inter-distance are always greater than a user-defined threshold) and the feasibility of the attitude control (always less than the maximum available torque) are taken into account.

The orbital dynamics takes into account the gravitational harmonics $\boldsymbol{\phi}_{i,j}$ up to degree and order $n_g = 20$ (values consistent with what reported in [157]), the third body perturbation of the Sun and the Moon, $\boldsymbol{p}_{\odot,\oplus}$ and $\boldsymbol{p}_{\leftmoon,\oplus}$, the drag $\boldsymbol{p}_D$ and the solar radiation pressure $\boldsymbol{p}_{SRP}$ perturbations are taken into account. The attitude

dynamics is controlled by $\boldsymbol{u}$ and the perturbative effect of the gravity gradient (GG) torque $\boldsymbol{M}_{GG}(\boldsymbol{r}, \boldsymbol{\xi})$ is modeled.

The attitude re-orientation problem has been stated as an *inverse dynamics* subproblem inside the generic problem in Eq. (8.2). In fact the angular velocities of the satellites and the torques required for the attitude maneuvers are obtained as a function of the attitude histories. Major details are given in Sec. 8.5.1. Note that when considering the atmospheric drag and the solar radiation pressure perturbations, the attitude and the orbital dynamics are coupled. The control constraint is required in order to deal with feasible control laws. The path constraint imposes a minimum safety inter-distance ∂_{min} among all the satellites during the entire maneuver in order to ensure the collision avoidance.

8.3 Reference relative trajectories

In this work, circular and near circular orbits will be taken into account. Let us consider the chief satellite (subscripted with c) centered at the origin of the rotating Euler–Hill frame $\mathscr{L} = \{\boldsymbol{X}_{\mathscr{L}}, \boldsymbol{Y}_{\mathscr{L}}, \boldsymbol{Z}_{\mathscr{L}}\}$ as in Fig. 8.1. Some reference trajectories of the deputy satellite (subscripted with d and identified by the position vector $\boldsymbol{\rho}_d = [x_d, y_d, z_d]$) are defined in $\mathscr{L}$ by means of the linear model described by the HCW equations [131],

$$\begin{aligned} \ddot{x}_d - 2n_c\dot{y}_d - 3n_c^2 x_d &= 0\,, \\ \ddot{y}_d + 2n_c\dot{x}_d &= 0\,, \\ \ddot{z}_d + n_c^2 z_d &= 0\,, \end{aligned} \tag{8.3}$$

where $n_c = (\mu/a_c^3)^{1/2}$, the mean motion, is supposed to be constant. For a close relative trajectory, we impose the *energy matching condition* [132] given by $\delta a = a_d - a_c = 0$. Let us define the Relative Orbit Elements (ROE) [158],

$$\delta\boldsymbol{\alpha} = \begin{bmatrix} \delta a \\ \delta\lambda \\ \delta e_x \\ \delta e_y \\ \delta i_x \\ \delta i_y \end{bmatrix} = \begin{bmatrix} a_c^{-1}(a_d - a_c) \\ u_d - u_c + (\Omega_d - \Omega_c)\cos(i_c) \\ e_d\cos(\omega_d) - e_c\cos(\omega_c) \\ e_d\sin(\omega_d) - e_c\sin(\omega_c) \\ i_d - i_c \\ (\Omega_d - \Omega_c)\sin(i_c) \end{bmatrix}, \tag{8.4}$$

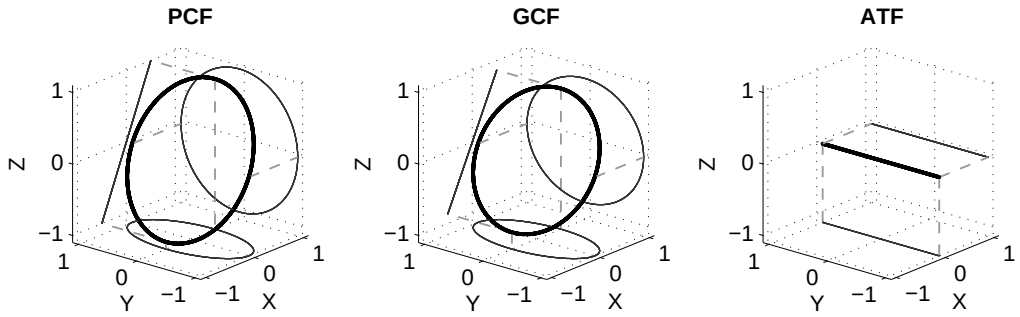


Fig. 8.2: Reference relative trajectories.

where $\mathbf{æ} = [a, e, i, \Omega, \omega, M]$ are the classical orbital elements and $u = M + \omega$. Accordingly, the trajectory generating from (8.3) can be expressed as

$$\begin{aligned} x_d(t) &= -a_c \delta e \cos\left(n_c t - \varphi\right), \\ y_d(t) &= a_c \delta\lambda + 2a_c \delta e \sin\left(n_c t - \varphi\right), \\ z_d(t) &= a_c \delta i \sin\left(n_c t - \theta\right), \end{aligned} \tag{8.5}$$

where

$$\begin{aligned} \delta e &= (\delta e_x^2 + \delta e_y^2)^{1/2}, \\ \delta i &= (\delta i_x^2 + \delta i_y^2)^{1/2}, \\ \varphi &= \tan^{-1}(\delta e_y / \delta e_x), \\ \theta &= \tan^{-1}(\delta i_y / \delta i_x). \end{aligned} \tag{8.6}$$

Eq. (8.5) is used to define the geometrical constraints of the initial and final configurations. Note that Eq. (8.5) is the same as Eq. (4) in [8], here reported for the sake of completness

$$\begin{aligned} x_d(t) &= -a_c \delta e \cos\left(n_c t - \delta\omega\right), \\ y_d(t) &= a_c \left(\delta\omega + \delta M + \cos i \delta\Omega\right) + 2a_c \delta e \sin\left(n_c t - \delta\omega\right), \\ z_d(t) &= a_c \left(-\delta\Omega \sin i_c \cos n_c t + \delta i \sin n_c t\right), \end{aligned} \tag{8.7}$$

which was written as a function of the differential orbital elements

$$\delta\mathbf{æ} = [\delta a, \delta e, \delta i, \delta\Omega, \delta\omega, \delta M] \tag{8.8}$$

and was valid for $e_c = 0$ or $e_c \to 0$ and $\delta\omega_c \to 0$ (the sign of $\delta\omega$ in Eq. (4) of Ref. [8] is $-$ instead of $+$, and δa is zero). With reference to Fig. 8.2, the following three formations are defined:

1. *Along-Track Formation* (ATF): the satellites are on the same orbit with small differences in anomaly. Consequently, only $\delta\lambda \neq 0$, while all the other ROE are

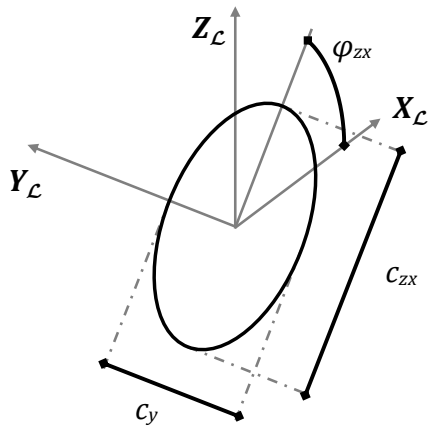


Fig. 8.3: Dimension and inclination of the relative motion ellipse.

zero. Defining $y_0 = a_0\delta\lambda$, the formation is then described by $[x(t),\, y(t),\, z(t)] = [0,\, y_0,\, 0]$.

2. *General Circular Formation* (GCF): the relative trajectory lies on a plane and is circular with radius R, centered in the position of the *chief* satellite.

3. *Projected Circular Formation* (PCF): the relative trajectory lies on a plane, is elliptical and the yz projection is a circle, centered in the position of the *chief* satellite and with radius R.

Close relative trajectories can be represented in Cartesian coordinates in $\mathscr{L}$ as

$$\begin{aligned} x_d(t) &= 0.5\,R\sin\left(n_c t + \alpha_0\right) = a_c\delta e\sin\left(n_c t + \alpha_0\right)\,, \\ y_d(t) &= y_0 + R\cos\left(n_c t + \alpha_0\right) = a_c\delta\lambda + 2a_c\delta e\cos\left(n_c t + \alpha_0\right)\,, \\ z_d(t) &= 0.5\,K_{zx}R\sin\left(n_c t + \alpha_0\right) = a_c\delta i\sin\left(n_c t + \alpha_0\right)\,, \end{aligned} \tag{8.9}$$

where R is related to the dimension of the formation ellipse and α_0 is the initial phase angle. Comparing Eq. (8.5) and Eq. (8.9), given that $\sin(\alpha) = -\cos(\alpha + \pi/2)$ and $\cos(\alpha) = \sin(\alpha + \pi/2)\ \forall\alpha \in \mathbb{R}$, the following equalities have been established,

$$-\varphi = \alpha_0 + \frac{\pi}{2}\,,\ \theta = -\alpha_0. \tag{8.10}$$

GCF, PCF and ATF are then obtained as the following special case:

- GCF: $K_{zx} = \sqrt{3}$ (centered in the origin of $\mathscr{L}$ if $y_0 = 0$).
- PCF: $K_{zx} = 2$ (centered in the origin of $\mathscr{L}$ if $y_0 = 0$).
- ATF: $R = 0$ and $y_0 \neq 0$.

In the general case, the xy-projection of Eq. (8.5) and (8.9) is a $2:1$ ellipse centered at $(0, y_0)$ (see Fig. 8.8 for clarity). With reference to Fig. 8.3, the ellipse axes c_y and c_{zx} are

$$c_y = 2R\,,\ c_{zx} = R\sqrt{K_{zx}^2 + 1} \tag{8.11}$$

and the inclination of the relative motion plane is

$$\varphi_{zx} = \tan^{-1} K_{zx}. \tag{8.12}$$

The mathematical models for close relative motion given in Eqs. (8.5), (8.7), (8.9) are linear approximations valid when no perturbations are considered. Actually, more accurate models exist, as the one described in [159] where the ROE formalism in employed to introduce the J_2 gravitational contribution and the differential drag. However, employing such models, the simple and intuitive closed form solutions presented so far to define the PCF, GCF and ATF configurations cannot be easily recognized.

When introducing non-linearities, we can only impose such reference orbits at the $t = t_0$. As described in [132], initial conditions on δa may be modified to take into account the along-track drift introduced by the J_2, which represents one of the most significant non-keplerian effects.

8.4 Perturbations as control inputs

In this paragraph, the opportunity to perform perturbation-based maneuvers is introduced. First, in Sec. 8.4.1, the mathematical description of the drag and solar radiation perturbations is given. Second, in Sec. 8.4.2, a preliminary feasibility analysis about the opportunity to use perturbations as control inputs is performed. Then, in Sec. 8.4.3, the general features of perturbation-based maneuvers are highlighted, while in Sec. 8.4.4 the characteristics of minimum-time maneuvers with in-plane forces are described.

8.4.1 Perturbations introduced by natural forces

The orbital dynamics of the satellites are modeled as in [8]. From the analysis reported in [160], the lift/drag ratio is related to the thermal accommodation coefficient and attains a maximum value of 0.05 when a diffuse re-emission model is employed with complete accommodation. In more recent studies, as in [161], an accommodation coefficient close to 1 is recommended for circular orbit below 500 km. As a consequence, in this work it is assumed that the lift component of the atmospheric force is negligible with respect to drag.

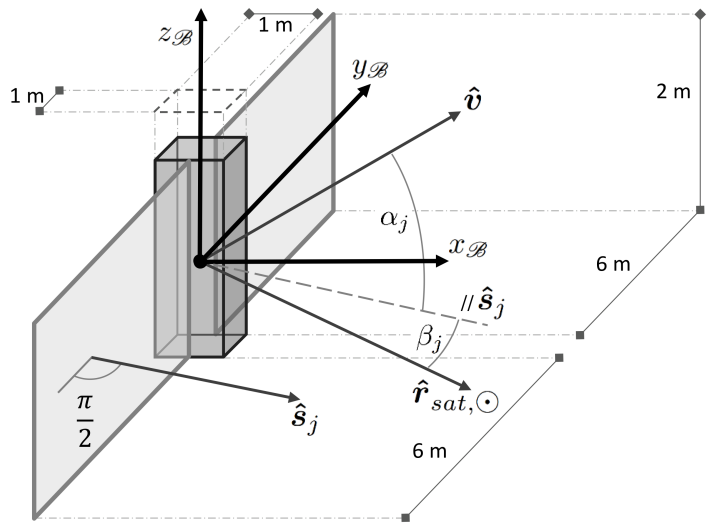


Fig. 8.4: Satellite dimensions, surfaces and orientation angles.

With reference to Fig. 8.4, where the body reference frame $\mathscr{B} = \{x_{\mathscr{B}}, y_{\mathscr{B}}, z_{\mathscr{B}}\}$ is shown, the drag perturbation $\boldsymbol{p}_D$ is given by [162]

$$\boldsymbol{p}_D = \boldsymbol{p}_D(\boldsymbol{r}, \boldsymbol{\xi}, t) = -\frac{\varrho(\boldsymbol{r}, t)\, C_D}{2m} \sum_{j:\alpha_j < \frac{\pi}{2}} (S_j \cos\alpha_j(\boldsymbol{\xi}))v^2 \hat{\boldsymbol{v}} \tag{8.13}$$

where C_D is the drag coefficient, m is the mass of the satellite and

$$\alpha_j(\boldsymbol{\xi}) = \cos^{-1}\left(\hat{\boldsymbol{s}}_j(\boldsymbol{\xi}) \cdot \hat{\boldsymbol{v}}\right) . \tag{8.14}$$

The unit vectors $\hat{\boldsymbol{s}}_j(\xi)$ and $\hat{\boldsymbol{v}}$ represent the normal to each j^{th} surface S_j of the satellite and the normalized velocity of the satellite, respectively. The density ϱ in Eq. (8.13) is provided by the Jacchia-Roberts model [157], which is a good compromise between precision and implementation complexity. In fact, uncertainties in the density estimation still remain also in more sophisticated models [163] thus not justifying the implementation of more complicated algorithms in this work. The acceleration components due to a non-fixed atmosphere are neglected since they would not significantly affect the results.

Similarly to the case of Eq. (8.14), the angle β_j in Fig. 8.4 that each surface makes with the solar radiation flux direction $\hat{\boldsymbol{r}}_{sat,\odot}$ is given by

$$\beta_j(\boldsymbol{\xi}) = \cos^{-1}\left(\hat{\boldsymbol{s}}_j(\boldsymbol{\xi}) \cdot \hat{\boldsymbol{r}}_{sat,\odot}\right) \tag{8.15}$$

where $\hat{\boldsymbol{r}}_{sat,\odot}$ is a unit vector directed from the satellite to the Sun. Considering an ideal reflecting surface, the SRP perturbation $\boldsymbol{p}_{SRP}$ is modeled as [162]

$$\boldsymbol{p}_{SRP} = \boldsymbol{p}_{SRP}(\boldsymbol{r}, \boldsymbol{\xi}, t) = 2\zeta(t)\gamma \frac{\mathcal{S}_0}{mc} \frac{\|\boldsymbol{r}_{\oplus,\odot}\|}{\|\boldsymbol{r}_{\oplus,\odot} - \boldsymbol{r}\|} \sum_{j:\beta_j < \frac{\pi}{2}} \Big(S_j \cos^2(\beta_j)\, \hat{\boldsymbol{s}}_j \Big) \tag{8.16}$$

where γ is the reflectivity coefficient, $\mathcal{S}_0 = 1352.098$ kg/s^2, $c = 2.988 \cdot 10^5$ km/s. The shadow function $\zeta(t)$ is equal to 1 when the satellite is outside the Earth shadow cone, equal to 0 when it is inside and equal to 0.5 in penumbra. As reported in [162], the solar radiation pressure contribution is not as strong as the drag contribution below altitudes of about 800 km.

Differential accelerations are defined as vectorial differences between perturbation accelerations acting on two distinct spacecrafts in the formation. Hence, DD acceleration $\boldsymbol{a}_{DD}$ and differential solar radiation pressure (DSRP) acceleration $\boldsymbol{a}_{DSRP}$ acting on the i-th satellite with respect to the j-th satellite are

$$\begin{aligned} \boldsymbol{a}_{DD}^{(i,j)} &= \boldsymbol{p}_D(\boldsymbol{r}_i, \boldsymbol{\xi}_i, t) - \boldsymbol{p}_D(\boldsymbol{r}_j, \boldsymbol{\xi}_j, t), \\ \boldsymbol{a}_{DSRP}^{(i,j)} &= \boldsymbol{p}_{SRP}(\boldsymbol{r}_i, \boldsymbol{\xi}_i, t) - \boldsymbol{p}_{SRP}(\boldsymbol{r}_j, \boldsymbol{\xi}_j, t), \end{aligned} \tag{8.17}$$

where $i, j \in \{1, ..., N_{\mathcal{F}}\}$, $i \neq j$. Perturbation accelerations may be used as control inputs for the SFF reconfiguration illustrated by the well known GVE (see [162, 164] for detailed discussion of GVE). We will express the GVE for circular orbits and express the perturbation as $\boldsymbol{p} = [p_n, p_t, p_h]$ where p_n is the in-plane component perpendicular to the orbit, p_t is the in-plane component tangential to the orbit and p_h is the out-of-plane component. As a consequence, the classical orbital elements satisfies the following differential equations:

$$\begin{aligned} \frac{d\Omega}{dt} &= \frac{r \sin\theta}{h \sin i} p_h, \\ \frac{di}{dt} &= \frac{r \cos\theta}{h} p_h, \\ \frac{d\omega}{dt} &= \frac{1}{ev}\left(2\sin\nu\, p_t + \left(2e + \frac{r}{a}\cos\nu \right) p_n \right) - \frac{r \sin\theta \cos i}{h \sin i} p_h, \\ \frac{da}{dt} &= \frac{2a^2 v}{\mu} p_t, \\ \frac{de}{dt} &= \frac{1}{v}\left(2\Big(e + \cos\nu\Big) p_t - \left(\frac{r}{a}\sin\nu \right) p_n \right), \\ \frac{dM}{dt} &= n - \frac{b}{eav}\left(2\sin\nu\Big(1 + \frac{e^2 r}{p}\Big) p_t + \left(\frac{r}{a}\cos\nu \right) p_n \right), \end{aligned} \tag{8.18}$$

where $\theta = \omega + \nu$. Actually, it should be noted that Eq. (8.18) become singular when considering $e = 0$ or $i = 0$. In such cases, GVE may be rewritten considering equinoctial orbital elements. However, Eq. (8.18) gives the proper information to

understand what can be obtained using as control input the natural perturbation forces. Looking at Eq. (8.13), it can be seen that the drag force is directed along $-\hat{\boldsymbol{v}}$ which may be confused with $-Y_{\mathscr{L}}$ for very small eccentricities. As a consequence, the drag force may only be used to modify a, e, ω and M. With the hypothesis of negligible lift, the drag force is a dissipative force and, looking at Eq. (8.4) and (8.5), only the in-plane x and y components may be controlled (i.e., δa, $\delta\lambda$, δe_x and δe_y). Conversely, the SRP has both in-plane and out-of-plane effects, as stated in Eq. (8.16), giving the possibility to vary all the orbital parameters. However, the intensity of the force depends on the angle $\beta(\boldsymbol{\xi})$. In this case, a key role is played by the orientation of the chief orbital plane, as it can vary the intensity of the in-plane and out-of-plane components of the SRP depending on the position of the Sun in $\mathscr{I}$. In fact, for an equatorial orbit the maximum value of the SRP is attained when $\beta = 23^\circ\,27'$, i.e. when the force is in the Ecliptic plane, thus having the in-plane component much bigger than the out-of-plane component. Conversely, in a polar orbit the maximum attainable magnitudes of in-plane and out-of-plane components are switched.

8.4.2 Preliminary feasibility study of perturbation-based maneuvers

The proposed maneuvering approach is based on the distinction between absolute perturbations and differential perturbations. The former are those forces that are common to all the spacecrafts in the formation and only depend on satellite position and environment (e.g. gravitational harmonics) whereas the latter are forces that can be modified with regard to magnitude and/or direction. Absolute perturbations can make unstable the PCF, GCF and ATF configurations or impact on the guidance while performing a maneuver. For instance, the J_2 perturbation sensibly modifies the cross-track relative motion and the along-track separation if the inclinations of the orbits are different, as shown in [159]. Atmospheric drag and solar radiation pressure are differential perturbations that can be used to perform reconfiguration maneuvers only if they can counteract the other orbital perturbations affecting the formation.

An interesting graph comparing the magnitudes of all the perturbation for all the relevant orbital regimes is reported in [157]. In LEO, the most relevant perturbation is the one due to the J_2. A preliminary analysis of the intensity of the gravity gradient acceleration $\nabla \boldsymbol{g}_{J_2}$ due to J_2 can be obtained using the linear approximation valid for near-circular orbits reported in [132],

$$\nabla \boldsymbol{g}_{J_2} = \kappa \boldsymbol{\Lambda} \boldsymbol{\rho}_d \tag{8.19}$$

where

$$\boldsymbol{\Lambda} = \begin{bmatrix} 1-3\mathrm{s}_{i_c}^2\mathrm{s}_{\theta_c}^2 & \mathrm{s}_{i_c}^2\mathrm{s}_{2\theta_c} & \mathrm{s}_{2i_c}\mathrm{s}_{\theta_c} \\ \mathrm{s}_{i_c}^2\mathrm{s}_{2\theta_c} & \mathrm{s}_{i_c}^2\left(\frac{7}{4}\mathrm{s}_{\theta_c}^2-\frac{1}{2}\right)-\frac{1}{4} & -\frac{1}{4}\mathrm{s}_{2i_c}\mathrm{c}_{\theta_c} \\ \mathrm{s}_{2i_c}\mathrm{s}_{\theta_c} & -\frac{1}{4}\mathrm{s}_{2i_c}\mathrm{c}_{\theta_c} & \mathrm{s}_{i_c}^2\left(\frac{5}{4}\mathrm{s}_{\theta_c}^2+\frac{1}{2}\right)-\frac{3}{4} \end{bmatrix}, \tag{8.20}$$
$$\kappa = 6J_2\left(\frac{\mu R_e^2}{r_c^5}\right),\ \mathrm{s}_{(\cdot)} = \sin(\cdot),\ \mathrm{c}_{(\cdot)} = \cos(\cdot).$$

In Eq. (8.20), θ_c is the true anomaly of the chief spacecraft and $J_2 = 1.082 \times 10^{-3}$. An upper bound of the magnitude of $\nabla \boldsymbol{g}_{J_2}$ can be obtained employing the property (valid for every general or induced norm [165])

$$\|\Lambda \boldsymbol{\rho}_d\| \leq \|\Lambda\| \|\boldsymbol{\rho}_d\|. \tag{8.21}$$

Considering $\|\boldsymbol{\rho}_d\| = 1$ km and using the 2-norm to evaluate $\|\boldsymbol{\Lambda}\|$, the maximum magnitude of the J_2 gradient acceleration can be found considering $\theta_c \in [0, 2\pi]$ for each inclination, i.e.

$$\max(\|\nabla \boldsymbol{g}_{J_2}\|) = \max_{\|\boldsymbol{\rho}_d\|=1\,\mathrm{km}} (\|\nabla \boldsymbol{g}_{J_2}(i_c, r_c)\|) = \kappa(r_c) \max_{\theta_c \in [0,2\pi]} \|\boldsymbol{\Lambda}(i_c, \theta_c)\|_2 \,. \tag{8.22}$$

A preliminary estimation of DD experienced by the formation can be obtained considering $C_D = 2.2$ and $m = 100$ kg for both the spacecrafts, same orbital velocities (in magnitude and direction) and minimum/maximum values of the atmospheric density from the Harris-Priester model, $\varrho_{HP}^{(min)}$ and $\varrho_{HP}^{(max)}$, shown in Table 8.2 (same density for both the spacecrafts). With these hypotheses, an upper bound of the DD magnitude can be estimated as

$$\|\boldsymbol{a}_{DD}\| = \frac{\varrho_{HP}\, C_D}{2m} v^2 \Delta S, \tag{8.23}$$

Tab. 8.2: Reference density values from the Harris-Priester model (from [162]).

Altitude (km)	Minimum density, $\varrho_{HP}^{(min)}$ (kg/m^3)	Maximum density, $\varrho_{HP}^{(max)}$ (kg/m^3)
200	2.557×10^{-10}	3.162×10^{-10}
300	1.708×10^{-11}	3.526×10^{-11}
400	2.249×10^{-12}	7.492×10^{-12}
500	3.916×10^{-13}	2.042×10^{-12}
600	8.070×10^{-14}	6.390×10^{-13}
700	2.043×10^{-14}	2.185×10^{-13}
800	7.069×10^{-15}	8.059×10^{-14}

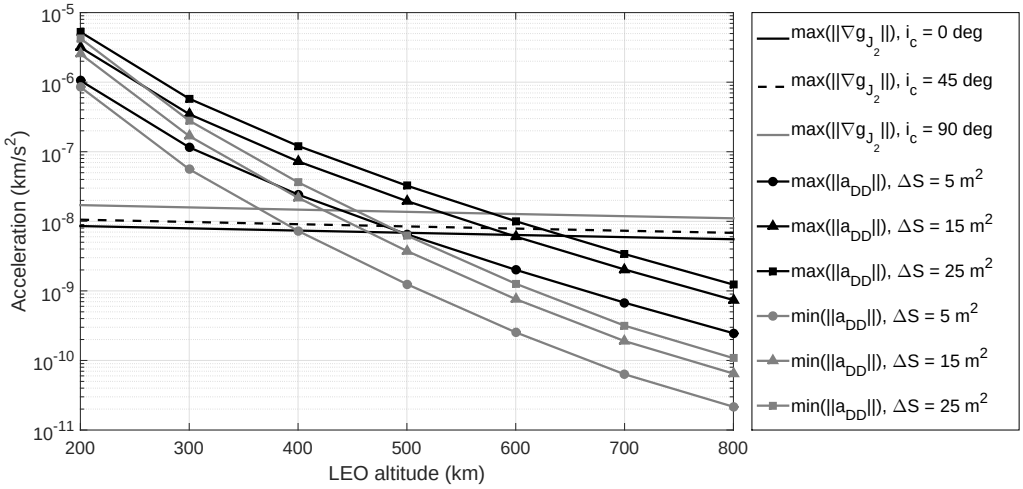


Fig. 8.5: Comparison between J_2 gravity gradient acceleration ($\|\boldsymbol{\rho}_d\| = 1$ km) and DD.

where ΔS is the difference between maximum and minimum exposed areas. Moreover, maximum and minimum values of $\|\boldsymbol{a}_{DD}\|$ as a function of ϱ_{HP} are

$$\max(\|\boldsymbol{a}_{DD}\|) = \|\boldsymbol{a}_{DD}(\varrho_{HP}^{(max)})\|\,,\ \min(\|\boldsymbol{a}_{DD}\|) = \|\boldsymbol{a}_{DD}(\varrho_{HP}^{(min)})\|. \tag{8.24}$$

In Fig. 8.5, a comparison of the magnitudes of J_2 gradient acceleration and DD perturbations is shown. The orbital inclinations i_c and the exposed area ΔS have been considered as parameters. As can be seen, the graph suggests that up to 400-500 km the magnitude of the DD perturbation is bigger or similar to the magnitude of $\nabla \boldsymbol{g}_{J_2}$ (up to $\|\boldsymbol{\rho}_d\| = 1$ km). Therefore, it is expected that reconfiguration maneuvers can be accomplished in low LEOs. At higher LEOs, the differential drag is less than $\nabla \boldsymbol{g}_{J_2}$, suggesting that reconfiguration maneuvers cannot be performed unless very large (and unreliable) exposed areas are considered. The opportunity to perform DD-based reconfiguration maneuvers increases with high vales of ΔS (i.e., no maneuvers can be accomplished with near-spherical satellites). Considering the exposed areas of the satellite model in Fig. 8.4, maneuvers can be reasonably performed up to 400-500 km.

Similar analysis can be performed in MEO and GEO, where other perturbations such as the J_{22} and the third body perturbation (from Sun and Moon) must be considered. However, such preliminary analyses can only give an approximated order of magnitude for the required exposed area, since the necessary model simplifications can affect the results. For example, the intensity of the J_2 perturbation varies with the latitude, and the in-plane and out-of-plane contributions can have helpful or disturbing effects in different points of the orbit. As a consequence, for some orbital regimes reconfiguration maneuvers might be accomplished even if the average DD magnitude is lower than the average magnitude of the J_2 gradient acceleration.

The proposed numerical approach is then a useful instrument when a more reliable feasibility analysis is required to estimate the opportunity to perform perturbation-based reconfigurations.

8.4.3 Insights on perturbation-based maneuvers

For close relative trajectories described by Eq. (8.9), a relationship between trajectory parameters R, K_{zx} and ROE can be found as

$$R = 2a_c\delta e\,,\ K_{zx} = \frac{z_d}{x_d} = \frac{\delta i}{\delta e}. \tag{8.25}$$

Variations of R influence all the components of the deputy motion but do not change the plane of the relative motion, i.e. the ratio z_d/x_d or φ_{zx} in Fig. 8.3. From Eq. (8.25), a variation of R can be induced by a variation of a_c (also a_d must change to preserve the energy matching condition) or a variation of δe. The inclination of the relative motion plane is dictated by the value of K_{zx}. However, K_{zx} also influences the dimension of the relative trajectory as it modifies the ellipse axis c_{zx} as stated in Eq. (8.11), stretching the trajectory along the $\boldsymbol{Z}_{\mathscr{L}}$ direction. As a consequence, both R and K_{zx} can affect, in different ways, the dimension of the relative ellipse. As a result of the GVE [164], out-of-plane forces can only modify i and Ω, in-plane forces can modify a, e and M while ω can change with a force with any direction. Since the ROE set is a combination of the orbital elements, effects of perturbations on the relative motion change depending on the direction of the perturbation forces.

With reference to Fig. 8.6, and considering out-of-plane forces, the following maneuvers can be accomplished:

- The dimension of the relative motion ellipse can be increased, as shown in Fig. 8.6(a), applying out-of-plane forces to the deputies. In this way, the trajectory geometry is modified as only one axis is affected (see Eq. (8.11)).

- The inclinations of the deputy satellites can be modified, increasing or decreasing the term $a\delta i$ in the z_d component of Eq. (8.5). Affecting the term δi, this maneuver modifies K_{zx} (and φ_{zx}) as can be seen from Eq. (8.25) and as shown in Fig. 8.6(a).

In contrast to the previous case, using only along-track forces, the following maneuvers can be accomplished:

- The dimension of the relative motion ellipse may be increased with in-plane forces directed along $\boldsymbol{Y}_{\mathscr{L}}$ and applied to all the satellites, as shown in Fig. 8.6(b). In this way a_c is increased, as well as the semi-major axes of the deputy

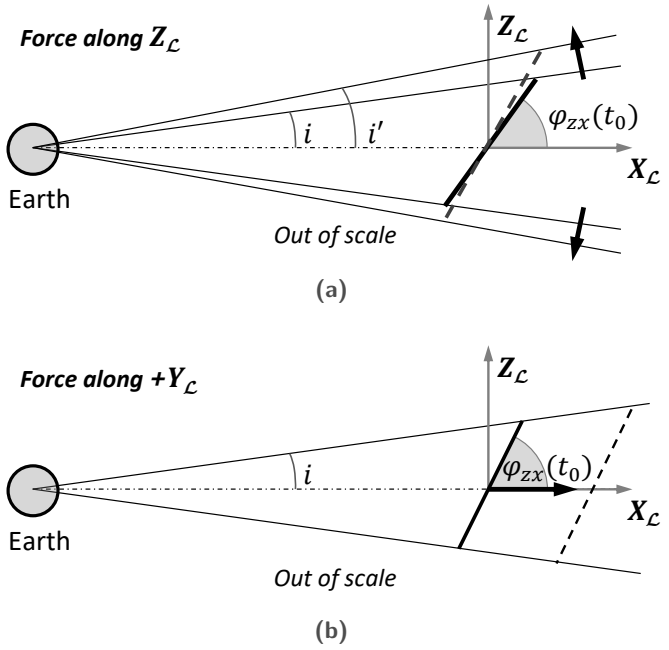


Fig. 8.6: Increasing the relative ellipse dimension with in-plane and out-of-plane forces.

satellites, enlarging the dimension of the relative motion ellipse. However, as opposed to the previous point, this maneuver does not modify the geometry of the relative ellipse, as both the ellipse axes are modified by the same amount (see Eq. (8.11)).

- Variations of the relative motion plane inclination can be obtained with in-plane forces. In fact, a variation of the angle φ_{zx} is obtained when the orbital parameter e (dictating the form of the absolute orbit) is changed. Such a maneuver affects δe and K_{zx} as stated in Eq. (8.25).

With regard to drag (only in-plane component) and solar radiation pressure (both in-plane and out-of-plane components), a summary of the maneuvering opportunities given by these perturbations is given below:

- As a consequence of the previous points, the drag force cannot increase R with constant K_{zx} since it is an in-plane force directed along $-\boldsymbol{Y}_{\mathcal{L}}$, opposite to the velocity direction. However, it may change the inclination of the relative motion plane, as shown in Fig. 8.7(a). All the formation satellites will decrease their value of the semi-major axis because of the dissipative effect of the drag. As stated in Sec. 8.4.1, the out-of-plane lift component is neglected.

- The SRP force might be used to increase the dimension of the relative motion ellipse, as an out-of-plane component can be obtained depending on the orbit geometry. Consequently, maneuvers as in Fig. 8.7(b) can be addressed with the SRP. Taking advantage of the in-plane and out-of-plane components of

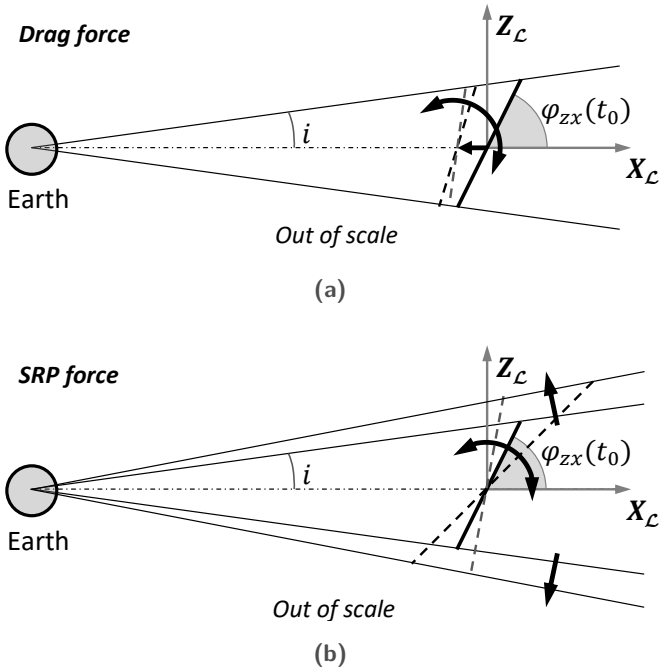


Fig. 8.7: Effects of drag and solar radiation pressure on the relative ellipse.

SRP, both R and K_{zx} can be properly controlled. In the results section, the SRP will be used to change K_{zx} and R modifying both the dimension and the inclination of relative trajectory. If the maneuver is properly planned, the final value of δe can be set to be equal to the initial value in order to modify R keeping K_{zx} constant. With the SRP, it is not expected to denote a semi-major axis decrease as in the case of the drag-induced maneuver.

8.4.4 Properties of minimum-time in-plane maneuvers

Let us consider maneuvers based on in-plane forces. Reconfiguration from PCF to GCF, or vice-versa, are considered. With reference to Fig. 8.8, let $z_{d,max}$ remain fixed during the reconfiguration maneuver, i.e.

$$\max(z_{d,PCF}) = \max(z_{d,GCF}). \tag{8.26}$$

This hypothesis is almost true for short-time maneuvers lasting about one orbit, and its validity will be verified by the reported simulations. The maximum z displacement for the PCF and GCF cases is given by

$$\max(z_{d,PCF}) = \max(R_{PCF}\sin(n_c t + \alpha_0)) = R_{PCF}\,, \tag{8.27}$$

$$\max(z_{d,GCF}) = \max\left(\frac{\sqrt{3}}{2}R_{GCF}\sin(n_c t + \alpha_0)\right) = \frac{\sqrt{3}}{2}R_{GCF}\,. \tag{8.28}$$

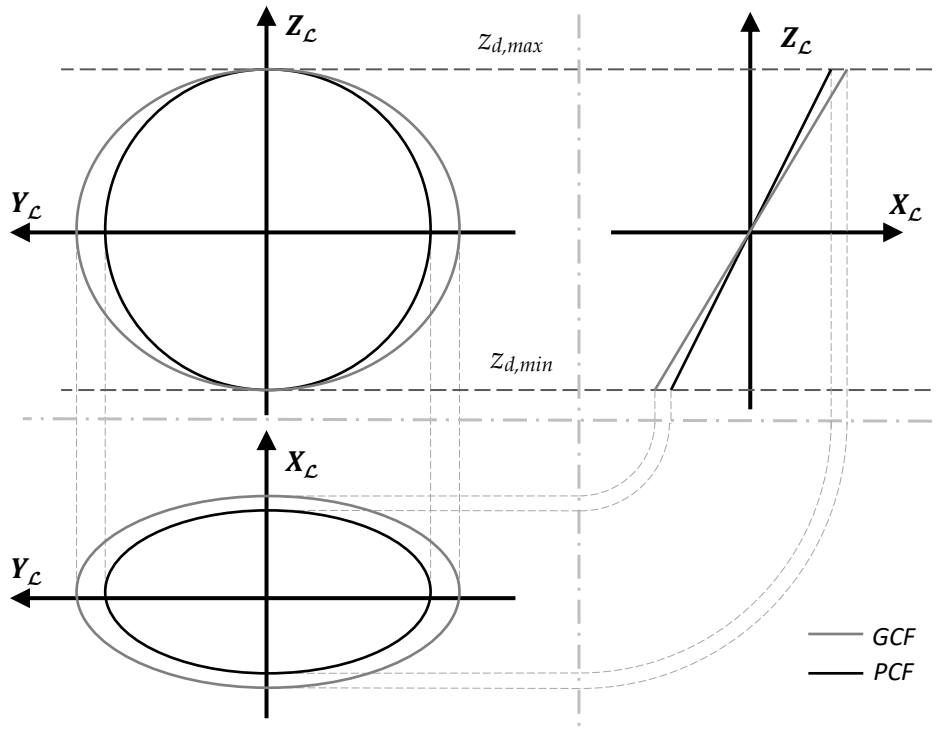


Fig. 8.8: Projection of PCF and GCF with same value of $z_{d,max}$.

Hence, considering a GCF to PCF, Eq. (8.26) leads to

$$R_{PCF}(t_f) = \frac{\sqrt{3}}{2} R_{GCF}(t_0) \approx 0.866\, R_{GCF}(t_0). \tag{8.29}$$

Conversely, for the opposite PCF to GCF maneuver,

$$R_{GCF}(t_f) = \frac{2}{\sqrt{3}} R_{PCF}(t_0) \approx 1.155\, R_{PCF}(t_0). \tag{8.30}$$

It is noteworthy that for the GCF to PCF maneuver the dimensions of the 2:1 ellipse in the xy plane have to be reduced. In the opposite case, the dimensions of the same ellipse must be increased.

8.5 Problem statement: IPSO strategy

In this section the optimal problem is transcribed. Some fundamental details are given to appreciate how the IPSO can be used and implemented. In Sec. 8.5.1 the IPSO is proposed to solve the attitude reorientation subproblem. In Sec. 8.5.2 whereas some technical details concerning the numerical implementation are given in Sec. 8.5.3.

8.5.1 IPSO for formation reconfiguration

The proposed technique is based on the IPSO. Knowing that the angular velocity may be expressed as a function of the attitude reprentation [114], i.e. $\boldsymbol{\omega} = \boldsymbol{g}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, \ddot{\boldsymbol{\xi}})$, then in the body reference frame $\mathcal{B}$ the torques are related to the kinematic by the Euler equation:

$$\Sigma \boldsymbol{M}_{ext} = I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} = \boldsymbol{f}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, \ddot{\boldsymbol{\xi}}) \,. \tag{8.31}$$

Once the attitude history is known along with its time derivatives, both the required torques and the angular velocity can be expressed in closed form. The most important consequences of the inverse approach are that: 1) the integration of the attitude dynamics is avoided, and 2) that the initial and final condition of the attitude kinematics may be imposed *a priori*. The proposed IPSO-based algorithm applies for the fully non-linear dynamics of the SFF. The improved IPSO technique is employed. In order to simplify the problem and the notation we will assume the attitude to have only 1 Degree of Freedom (DOF), i.e. $\xi_i(t) \in \mathbb{R}$. In the following, when 2 or 3 DOF will be needed, it will be explicitly reported. The proposed algorithm is based on the following steps: 1) every particle is associated to $N_{\mathcal{F}}$ attitude kinematic trajectories $\xi_i(t)$, $i = 1, 2, ..., N_{\mathcal{F}}$, defined in the time span $\{0, t_f\}$; 2) $\boldsymbol{p}_D(\xi)$ and $\boldsymbol{p}_{SRP}(\xi)$ are evaluated and the dynamics is integrated; 3) the final configuration is evaluated and compared with the required one; 4) the IPSO searches for the optimal solution which minimizes the maneuver time satisfying all the constraints reported in Eq. (8.2).

8.5.2 Overview of the optimization strategy

The whole optimization process is depicted in the block diagram shown in Fig. 8.9. The attitude reorientation problem has been stated as an inverse dynamics sub-problem inside the generic problem in Eq. (8.2).

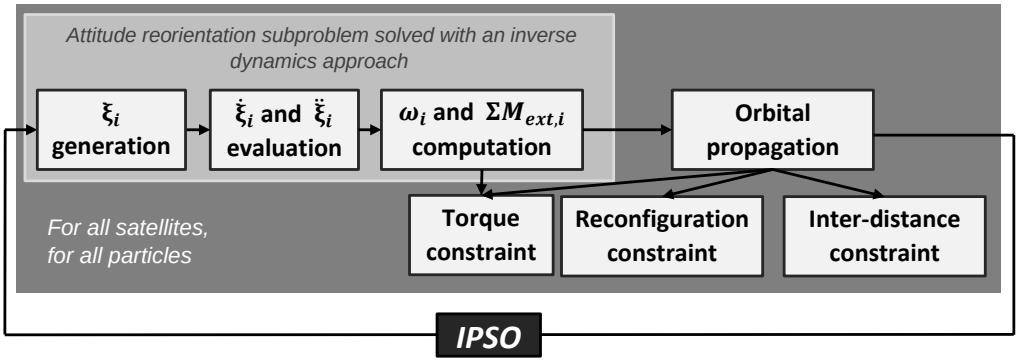


Fig. 8.9: Block diagram of IPSO applied to formation reconfiguration.

Using the improved B-spline approximation, we define the optimization parameters in the IPSO particles as

$$\mathbf{X} = [\tilde{\boldsymbol{\xi}}_1^T, \tilde{\boldsymbol{\tau}}_1^T, ..., \tilde{\boldsymbol{\xi}}_{N_\mathcal{F}}^T, \tilde{\boldsymbol{\tau}}_{N_\mathcal{F}}^T, t_f]^T \in \mathbb{R}^{2N_\mathcal{F}N_\mathcal{P}+1}, \tag{8.32}$$

where vectors $\tilde{\boldsymbol{\xi}}_i^T$ and $\tilde{\boldsymbol{\tau}}_i^T$ contains the optimization parameters ($N_\mathcal{P}$ for each vector) defining the B-spline curve for the attitude maneuvers of the satellites. As usually, once the attitude history is known along with its time derivatives, both the required torques and the angular velocity can be expressed in closed form. The most important consequences of the inverse approach are that: 1) the integration of the attitude dynamics is avoided, and 2) that the initial and final condition of the attitude kinematics may be imposed *a priori*. The attitude reorientation subproblem is reported in Fig. 8.9 by means of the first three blocks.

The attitude histories of the formation satellites are used to evaluate the drag acceleration, $\boldsymbol{p}_D(\boldsymbol{\xi})$, and the SRP acceleration, $\boldsymbol{p}_{SRP}(\boldsymbol{\xi})$, and the orbital dynamics are integrated. Once accomplished the integration task, the imposed constraints may be calculated. Finally, the formation configuration achieved after the maneuver is evaluated and compared with the required one. As depicted in Fig. 8.9, the IPSO searches for the optimal solution which minimizes the maneuver time satisfying all the imposed constraints reported in Sec. 8.2.

Note that the GG torque may be taken into account after the dynamics integration. In fact, we can evaluate the required control torque $\boldsymbol{u}$ as

$$\boldsymbol{u} = I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times I\boldsymbol{\omega} - \boldsymbol{M}_{GG}(\boldsymbol{\xi}, \boldsymbol{r})\,. \tag{8.33}$$

The performance index, the penalty functions and the descreasing tolerance technique are imposed following the criterions reported in Chapter 4. In Eq. (8.2) all the initial conditions are automatically satisfied, as well as the final conditions. In fact initial conditions on $\boldsymbol{X}$ are imposed by the numerical integration, whilst the initial and final conditions concerning the attitude in $\boldsymbol{Y}$ are imposed via the IPSO technique. As a consequence, all the attitude constraints are naturally satisfied thus simplifying the overall constraint handling strategy. Hence, only final conditions on $\boldsymbol{X}$ and the state/control path constraints must be taken into account in the optimization process and properly modelled in the performance index. The performance index J is designed as

$$J = J_0 + \mathcal{P} + \mu N_{viol}\,, \tag{8.34}$$

where the term $\mathcal{P}$ takes into account all the penalty functions associated to the constraints described in Sec. 8.2 and the last term is related to number of violated constraints. The term J_0 is designed in slightly different ways depending if the

maneuver is performed with the drag or the SRP. In the former case, $J_0 = t_f$. With the drag, no out-of-plane force is detected and the results are expected to confirm the speculations of Sec. 8.4.4 with regard to the value of $R(t_f)$. On the other hand, with the SRP the out-of-plane component can perturb the final result, influencing the value of the trajectory radius R. To choose a specific strategy consistent with the minimum-time maneuver we are searching for, $J_0 = t_f + \kappa|R(t_f) - R_f^*|$, where R_f^* assumes the values outlined in Sec. 8.4.4. Without the term in R, different maneuvers may be found with different values of $R(t_f)$, hence the proposed strategy allows to identify a unique maneuver. The same term may be used for another kind of maneuver, i.e. for imposing other values of the final radius of the relative trajectory. For instance, a GCF to GCF imposing $R(t_f) > R(t_0)$ will be presented with this regard. Obviouslly, intermediate cases can be searched for with combinations of the two strategies, which means that the SRP may be used to obtain slight variation of both K_{zx} and R.

With regard to $\mathcal{P}$, this term is given as the summation of the penalty functions associated to the boundary and path constraints to be satisfied. In further details,

$$\mathcal{P} = C + P + E_1 + E_2 \tag{8.35}$$

The terms C and P are the *control penalty function* and the *inter-link penalty function*, respectively associated to the control constraint c_1 and the path constraint p_1 of Eq. (8.2). They must be evaluated for all the N_T time instants $t_i \in \{t_0 = 0, t_1, ..., t_{N_T} = t_f\}$ where the numerical integration has been accomplished (see Sec. 8.5.3). The term C is defined as

$$C = \tilde{c} \sum_{j=1}^{N_{\mathcal{F}}} \sum_{k=1}^{3} \sum_{i=0}^{N_t} \xi_{jk}(t_i) \tag{8.36}$$

where $\tilde{c}$ is a user-defined constant and $\xi_{jk}(t_i)$ takes the following form:

$$\xi_{jk}(t_i) = \begin{cases} 0 & \text{if } \dfrac{|u_{j,k}(t_i)|}{u_{max}} - 1 < \Delta_c^{(k)} \\ 1 & \text{otherwise} \end{cases} \tag{8.37}$$

where $u_{j,k}$ is the value of the attitude control of the j^{th} along k^{th} body-axis. The tolerance $\Delta_c^{(k)}$ decreases during the simulation as it will be explained in 4.5.2. Similarly, P is defined as

$$P = \tilde{p} \sum_{j=1}^{N_{\mathcal{F}}-1} \sum_{k=j+1}^{N_{\mathcal{F}}} \sum_{i=0}^{N_T} \eta_{jk}(t_i) \tag{8.38}$$

where $\eta_{jk}(t_i)$ is:

$$\eta_{jk}(t_i) = \begin{cases} 0 & \text{if } \partial_{jk} = \|\boldsymbol{r}_j(t_i) - \boldsymbol{r}_k(t_i)\| > (1 - \Delta_p^{(k)})\partial_{min} \\ 1 & \text{otherwise} \end{cases} \tag{8.39}$$

The terms E_1 and E_2 associated to the final condition e_1 and e_2 of Eq.(8.2) might be defined as a function of the error between the state at the end of the integration and the desired state. However, it may be verified that even small errors in position and velocity at the final time may lead to considerable deviations after only one orbital period. In this sense, a more robust definition of E_1 and E_2 is required. Similarly to what has been done in [52], the idea is to compare the ideal relative state $[\boldsymbol{\rho}^{*T}, \dot{\boldsymbol{\rho}}^{*T}]^T$ given by the imposed final configuration with the relative state $[\boldsymbol{\rho}^T, \dot{\boldsymbol{\rho}}^T]^T$ generating from the final conditions of the integration. The comparison is carried out considering one orbit propagation and the linearized dynamics of Eq. (8.3). As a consequence, considering $t_i \in [t_f, t_f + T]$ with $T = 2\pi\sqrt{a^3/\mu}$, we define:

$$E_1 = \tilde{e}_1 \sum_{j=1}^{N_\mathcal{F}} \sum_{i=0}^{N_T} \psi_j(t_i) \qquad E_2 = \tilde{e}_2 \sum_{j=1}^{N_\mathcal{F}} \sum_{i=0}^{N_T} \zeta_j(t_i) \tag{8.40}$$

$$\psi_j(t_i) = \begin{cases} 0 & \text{if } \left\| \dfrac{\boldsymbol{\rho}_j(t_i)}{\bar{R}} - \dfrac{\boldsymbol{\rho}_j^*(t_i)}{R} \right\| < \Delta_r^{(k)} \\ 1 & \text{otherwise} \end{cases} \tag{8.41}$$

$$\zeta_j(t_i) = \begin{cases} 0 & \text{if } \left\| \dfrac{\dot{\boldsymbol{\rho}}_j(t_i)}{\bar{n}\bar{R}} - \dfrac{\dot{\boldsymbol{\rho}}_j^*(t_i)}{nR} \right\| < \Delta_v^{(k)} \\ 1 & \text{otherwise} \end{cases} \tag{8.42}$$

In Eq. (8.41) and (8.42) the quantities $\bar{R}$ and $\bar{n}$ are the average radius and mean motion of the obtained trajectory. Through the propagated orbit, the SFF parameters can be obtained and compared to the initial formation parameters. Note that $J = t_f$ when all the constraints are satisfied.

8.5.3 Implementation and orbital integration

The first dynamic constraint in Eq. (8.2) represents the center of mass dynamics which considers the most dominant perturbations. In addition to the drag and the solar radiation pressure perturbations, the gravitational harmonics up to degree and order 20 (values consistent with what reported in [157]) and the third body perturbation of the Sun and the Moon have been taken into account. The orbital integration of the satellites is carried out using a Gauss-Jackson scheme [166]. As shown in [157, 167], this method is faster than standard Runge-Kutta schemes and guarantees great numerical accuracy. Moreover, the Gauss-Jackson scheme is a fixed step-size integrator which allows a simple implementation of the input attitude

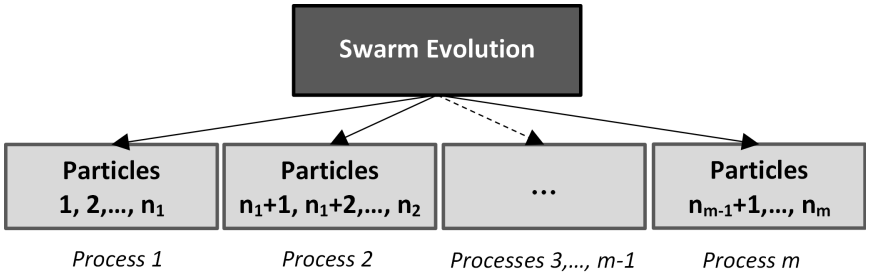


Fig. 8.10: Parallel computing associated to the PSO technique.

kinematics history inside the differential equation to be integrated. Note that the integration time instants must be equally spaced.

Even if the Gauss-Jackson integration scheme can help improving the performances of the integration, the computational effort is quite high because the integration requires the time-consuming evaluation of all the aforementioned perturbations. In order to make the numerical computation more efficient, the Parallel Computing utility available in Matlab® has been used. Accordingly, we make the swarm evolve dividing the particles among different processes where the individual bests may be updated. Conversely, the global and the local bests are computed only when all the particles of the current iteration have been evaluated.

The IPSO method is utilized here to solve a very hard problem which requires an efficient work-station in order to obtain the solution in reasonable time, in contrast to the prevoiously reported applications where the IPSO has been presented as an advantageous method for possible on-board implementation of an autonomous path planning.

8.6 Numerical Results

In this section some test cases are solved with the proposed technique. Such results extend the findings reported in [8], as more precise results are shown for difficult reconfiguration maneuvers. All the simulations have been carried out on a personal computer with an Intel®processor CoreTM i7-2670QM CPU @2.20GHz and with 6.00 GB of RAM.

8.6.1 Parameters and problem setting

A SFF with $N_{\mathcal{F}} = 4$ satellites is considered. Four different maneuvers are taken into account:

1. Modify a Projected Circular Formation into a General Circular Formation.

2. Modify a General Circular Formation into a Projected Circular Formation.

3. Increase the radius of a close relative formation.

4. Increase the interdistance of an Along-track formation.

From the previous discussions, it should be clear that case 3. can only be attempted using the SRP perturbation.

All the IPSO parameters have been chosen accordingly to Ref. [8], with the exception of some parameters that have been slightly modified (in particular, the maximum number of iterations has been increased to reach higher accuracies and the local model has been employed instead of the unified model). The swarm is set to have $N_{\mathcal{S}} = 30$ particles and the B-splines of degree 5 are defined with $N_{\mathcal{P}} = 10$ points. Weights and tolerances of the cost functions are defined as in Ref. [8].

With regard to the LEO case, results will be reported considering a single-axis maneuver since we have only one degree of freedom as the drag force always points in the opposite direction of the satellites velocity vector. However, instead of a yaw maneuver around $z_{\mathscr{B}}$ as in [8], here the maneuver is around $y_{\mathscr{B}}$ since the moment of inertia around this axis is lower than the one around $z_{\mathscr{B}}$. The result of the maneuver is not affected as the area exposed to the drag varies in a similar manner as in [8], but the control effort is reduced and a smaller value of the maximum torque can be chosen (see Table 8.3). When dealing with the solar radiation pressure in MEO and GEO, a two-axes maneuver around $y_{\mathscr{B}}$ and $z_{\mathscr{B}}$ is considered, as a rotation around $x_{\mathscr{B}}$ is useless in order to vary the exposed area. With $N_{\mathcal{F}} = 4$ and $N_{\mathcal{P}} = 10$ and referring to Eq. (8.32), the number of optimization variables is 81 of the one-axis maneuver and 121 for the two-axes maneuver.

All the relevant physical and geometrical properties of the four satellites are shown in Fig. 8.4 and in Table 8.3. The inertia tensor referred to the Body reference frame $\mathscr{B}$ of Fig. 8.4 is evaluated supposing a uniformly distributed mass over the central body and the solar panels.

Tab. 8.3: Satellites parameters.

Parameter	Value	Parameter	Value
m_{body} (kg)	90	$I_{x,\mathscr{B}}$ (kg·m^2)	$1.28e3$
m_{panel} (kg)	40	$I_{y,\mathscr{B}}$ (kg·m^2)	$6.41e1$
C_D	2.2	$I_{z,\mathscr{B}}$ (kg·m^2)	$1.23e3$
γ	1.5	M_{max} (Nm)	$1e$-3

Tab. 8.4: Initial conditions for the PCF-GCF maneuver.

	$\boldsymbol{r}_{\mathscr{L}}(t_0)$			$\dot{\boldsymbol{r}}_{\mathscr{L}}(t_0)$		
	x_d (km)	y_d (km)	z_d (km)	$\dot{x}_d$ (km/s)	$\dot{y}_d$ (km/s)	$\dot{z}_d$ (km/s)
			PCF to GCF maneuver			
SAT 1	0	0	0	0	0	0
SAT 2	0	1	0	$0.5n_c$	0	n_c
SAT 3	0.433	-0.500	0.866	$-0.25n_c$	$-0.866n_c$	$-0.5n_c$
SAT 4	-0.433	-0.500	-0.866	$-0.25n_c$	$0.866n_c$	$-0.5n_c$
			GCF to PCF maneuver			
SAT 1	0	0	0	0	0	0
SAT 2	0	1	0	$0.5n_c$	0	$0.866n_c$
SAT 3	0.433	-0.500	0.750	$-0.25n_c$	$-0.866n_c$	$-0.433n_c$
SAT 4	-0.433	-0.500	-0.750	$-0.25n_c$	$0.866n_c$	$-0.433n_c$

Results will be reported considering near-equatorial circular orbits in LEO and GEO and inclined circular orbits in MEO (the chief orbital parameters will be detailed in the following subsections). The initial condition for the PCF to GCF and GCF to PCF cases are reported in Table 8.4 expressed in $\mathscr{L}$ (velocities are a function of the mean motion, varying among LEO, MEO and GEO). PCF and GCF are considered with a central satellite (SAT1) and three follower satellites: SAT2 is aligned along the inertial velocity direction of the chief and SAT3 and SAT4 are equally spaced in the circular trajectory. The simulation epoch begins at the date January 2000, 00:00 a.m. UTC. For the LEO case, the initial and final attitudes are chosen with $x_{\mathscr{B}}$ aligned with $y_{\mathscr{L}}$ and $y_{\mathscr{B}}$ normal to the orbital plane. For the MEO and GEO cases, instead, $x_{\mathscr{B}}$ is aligned with the Sun-satellite direction and $y_{\mathscr{B}}$ is normal to the orbital plane. The attitude maneuvers in the LEO scenario are referred to the LVLH reference system $\mathscr{L}$, while in the MEO and GEO scenarios they are referred to the ECI system $\mathcal{I}$. In both cases, given the geometrical and structural symmetry of the satellite model, rotations are limited between -90 and +90 degree per axis.

The Gauss-Jackson integration [166] is carried out with a 60-seconds step and relative and absolute tolerances equal to 10^{-13} and 10^{-15}, respectively. Introducing $\varepsilon = 10^{-10}$, the IPSO stops according to the following convergence criterion reported in Sec. 4.5.3. A maximum number of 3000 iterations has been imposed in order to reasonably limit the required computational time.

8.6.2 Case 1: PCF-GCF reconfiguration in LEO

The chief orbital parameters are reported in Table 8.5. In [8], it has been shown that the higher LEOs do not allow to obtain good results when considering PCF to GCF maneuvers.

Tab. 8.5: Chief orbital parameters in LEO.

Parameter	Value	Parameter	Value
a (km)	6778	ω (deg)	0
e	0	i (deg)	1
Ω (deg)	10	M (deg)	0

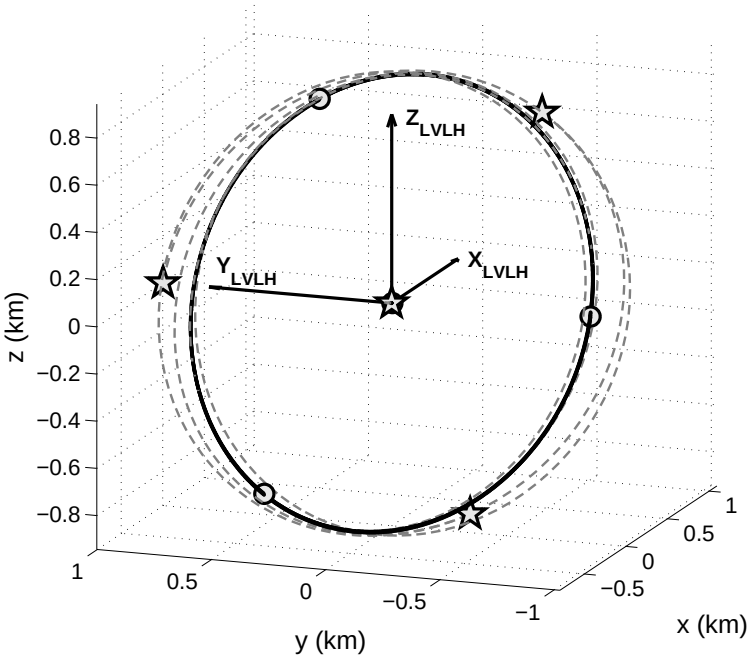


Fig. 8.11: GCF to PCF maneuver in the LVLH Coordinate System $\mathscr{L}$, in LEO.

First of all, let us look at Table 8.6 to understand the performances achievable with the drag perturbation. As can be seen, both the PCF to GCF and the GCF to PCF maneuvers may be accomplished with very good results. In both cases the time required for the maneuver is between one and one and a half orbits. The inclination of the relative orbit plane have been changed with great accuracy. Moreover, accordingly to the theoretical prediction stated in Sec. 8.4.4, the final GCF radius is 1.154 instead of 1.155 and the final PCF radius 0.866, as expected.

Tab. 8.6: Maneuver performances achievable in LEO with drag.

	$K_{zx}(t_0)$	$R(t_0)$		$K_{zx}(t_f)$		t_f	$\frac{R(t_f)}{R(t_0)}$
	(-)	(km)		(-)		(orbits)	(-)
	all deputies		SAT2	SAT3	SAT4	all deputies	
PCF to GCF	2.000	1.000	1.731	1.735	1.735	1.292	1.154
GCF to PCF	1.732	1.000	2.005	1.995	1.994	1.496	0.866

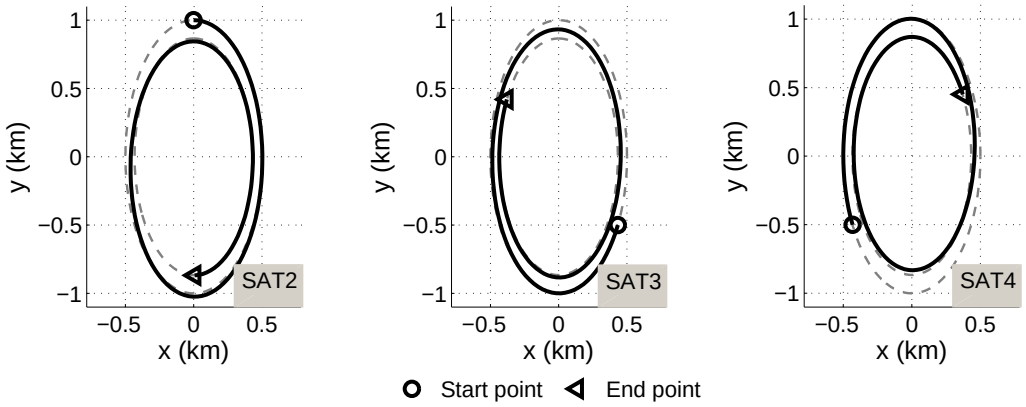


Fig. 8.12: GCF to PCF maneuver in the LVLH Coordinate System $\mathscr{L}$, xy projection in LEO.

This means that the maneuver time can be considered sufficiently short such that the hypothesis in Sec. 8.4.4 hold.

To get a greater insight into the proposed maneuver, we report some explanatory figures concerning the GCF to PCF maneuver (similar results are obtained vice-versa). In Fig. 8.11, the maneuver trajectory is shown in the $\mathscr{L}$ frame. The stars represent the initial positions of the satellites, while the circles are the final positions. The dashed lines are the optimal trajectories followed by the satellites and the solid lines represent one orbit propagation after the end of the maneuver. The trajectory is modified starting from a GCF (which is the outer trajectory) and arriving to a PCF (which is the inner trajectory) and the extremal values along the z direction are the kept constant (which also means that the radius of the GCF is quite the same as the semi-major axis of the finale relative ellipse given by the PCF). The angular distances among the satellites are not modified in a relevant way.

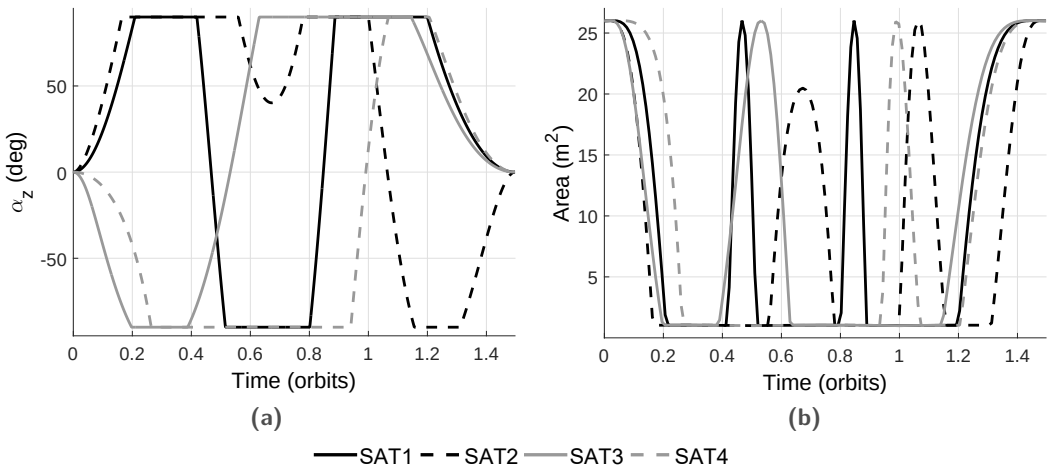


Fig. 8.13: Rotation angles (a) and exposed areas (b) for GCF to PCF maneuver, in LEO.

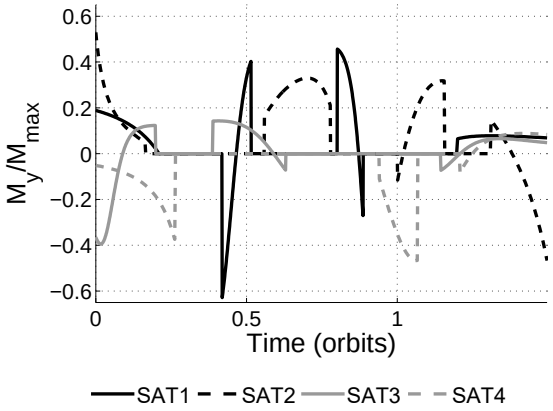


Fig. 8.14: History of the z-axis torque for the GCF to PCF maneuver, in LEO.

Fig. 8.12 reports the xy projection (in $\mathscr{L}$) of the maneuver proposed in Fig. 8.11. The three follower satellites start the transfer trajectory from the outside ellipse and finish their maneuver when they arrive on the inner ellipse. Note that, according to Eq. (8.9), the xy projection is always an ellipse with a y:x ratio equal to 2:1.

The rotation angles of the satellites about their y-axis are reported on the left in Fig. 8.13(a) while the adjacent right figure, Fig. 8.13(b), plots the resulting exposed area history. Given the symmetry of the considered satellite model, symmetrical reorientation attitude maneuvers are possible leading to the same reconfiguration results. Moreover, it should be noted that such attitude maneuvers are consistent with the maximum torque constraint, as can be seen from Fig. 8.14 where the normalized torques are reported. Therefore, it is numerically verified that the

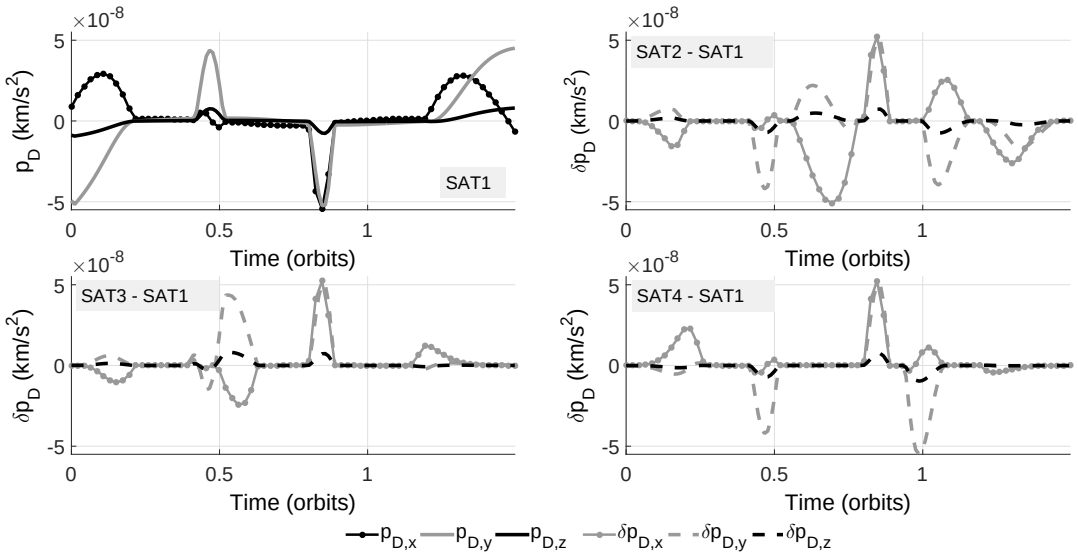


Fig. 8.15: Drag perturbation history for the GCF to PCF maneuver, in LEO.

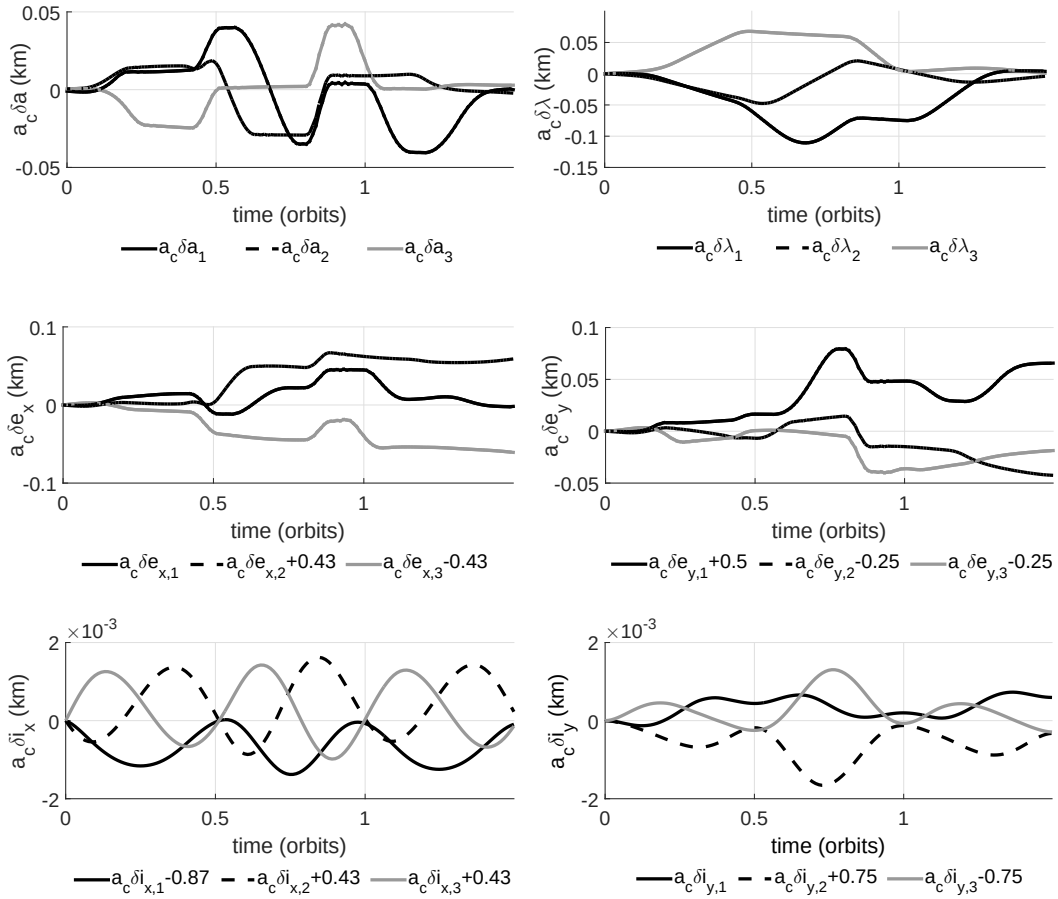


Fig. 8.16: ROE history for the GCF to PCF maneuver, in LEO (ROE components have been translated to zoom all the deputies' behavior).

disturbance effect of the GG torque does not affect the results, validating the results in this work and in [8].

One very interesting result reported in Fig. 8.13 is that the rotation, as well the exposed area, is likely to take only two values, i.e. the maximum exposed area or the minimum exposed area. This is particularly consistent with a minimum-time maneuver: if we were looking to the linearized model of relative motion with the HCW equations, it can be easily seen that reconfiguration maneuvers require a bang-bang control. Accordingly, the fact that the exposed area, which is proportional to the drag force, is mainly either minimum or maximum, suggests the near-optimality of the proposed solution. As a consequence of these attitude maneuvers, the drag force experienced by the satellites is reported in Fig. 8.15, where the force components are referred to $\mathscr{I}$. It can be seen that the order of magnitude of the drag force as well of the differential drag force is 10^{-8} km/s^2.

Tab. 8.7: Chief orbital parameters in MEO.

Parameter	Value	Parameter	Value
a (km)	26378	ω (deg)	0
e	0	i (deg)	60
$\{\Omega_1, \Omega_2\}$ (deg)	$\{280, 10\}$	M (deg)	0

Finally, it is noteworthy, looking at the ROE reported in Fig.8.16, the drag force only affects the in-plane elements, that is δa, $\delta\lambda$, δe_x and δe_y which is consistent with the GVE and the fact that the drag is in the direction of the velocity. The elements δi_x and δi_y show periodic variations due to the other conservative perturbations. It is noteworthy that δa and $\delta\lambda$ converges to a zero final vale, which is consistent with the expected results (see Eq. (8.5)).

8.6.3 Case 2: PCF-GCF reconfiguration in MEO

The chief orbital parameters for the MEO cases are reported in Table 8.7. The chief inclination has been set to 60 deg (value consistent with real MEO missions such as the GNSS constellations) to get the opportunity to obtain in-plane and out-of-plane components of the SRP with similar magnitudes. With this regard, two different values of ascending node have been chosen. With Ω_1, the line of nodes of the chief orbital plane is aligned to the Earth-Sun direction, hence the maximum component of the SRP is in plane. On the contrary, with Ω_2 the line of nodes is normal to the

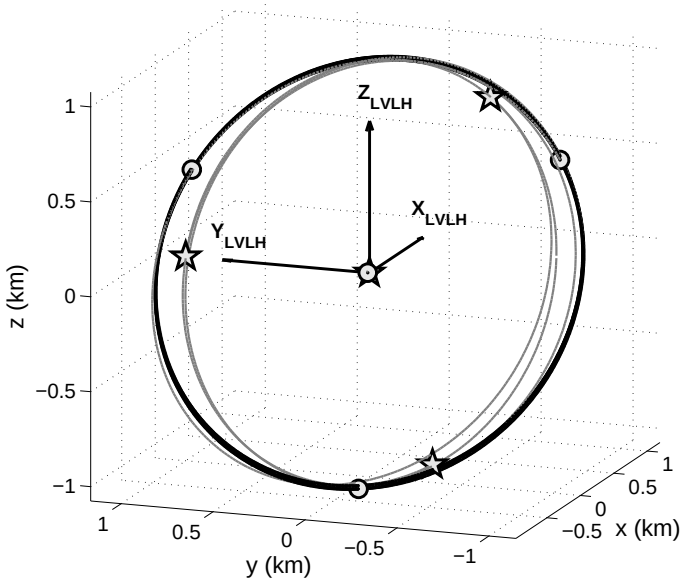


Fig. 8.17: PCF to GCF maneuver in the $\mathscr{L}$ Reference System in MEO with $\mathbf{\Omega_1}$.

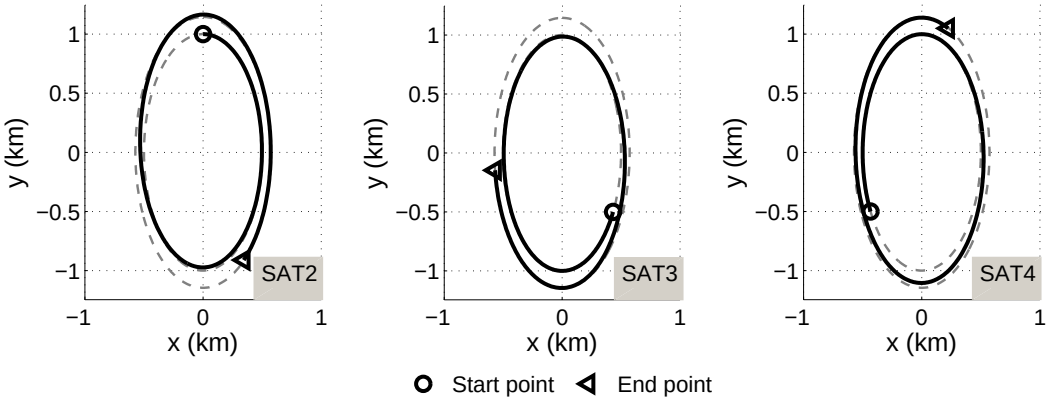


Fig. 8.18: PCF to GCF maneuver in the $\mathscr{L}$ Reference System, xy projection in MEO with Ω_1.

Earth-Sun direction and consequently the out-of-plane component of the SRP is comparable with the in-plane one.

A PCF to GCF maneuver performed with Ω_1 is depicted in Fig. 8.17 and Fig. 8.18. In this case we are moving from the inner relative trajectory to the outer one and the figures allow the reader to appreciate the ability to perform the maneuver with the SRP. From Fig. 8.17 it can be see that the maximum and minimum values along $z_{\mathscr{L}}$ remain quite constant satisfying the constraint imposed in the performance index.

Results reported in Table 8.8 for Ω_1 and Ω_2 show that in both the cases a good reconfiguration maneuvers may be achieved. The K_{zx} coefficients change achieving nice approximations of the expected values. The final radius is quite close to the one expected when using only the in-plane forces. Moreover, the time required for the cases with Ω_1 is greater than the time required when using Ω_2. This is consistent with

Tab. 8.8: Maneuver performances achievable with SRP in MEO.

	$K_{zx}(t_0)$	$R(t_0)$	$K_{zx}(t_f)$			t_f	$\frac{R(t_f)}{R(t_0)}$
	(-)	(km)		(-)		(orbits)	(-)
	all deputies		SAT2	SAT3	SAT4	all deputies	
	$i = 60$ deg, line of nodes along Earth-Sun direction						
PCF to GCF	2.000	1.000	1.723	1.749	1.745	1.400	1.146
GCF to PCF	1.732	1.000	2.003	2.001	2.002	1.211	0.872
	$i = 60$ deg, line of nodes normal to Earth-Sun direction						
PCF to GCF	2.000	1.000	1.753	1.733	1.732	1.015	1.138
GCF to PCF	1.732	1.000	1.999	2.000	2.000	1.008	0.874

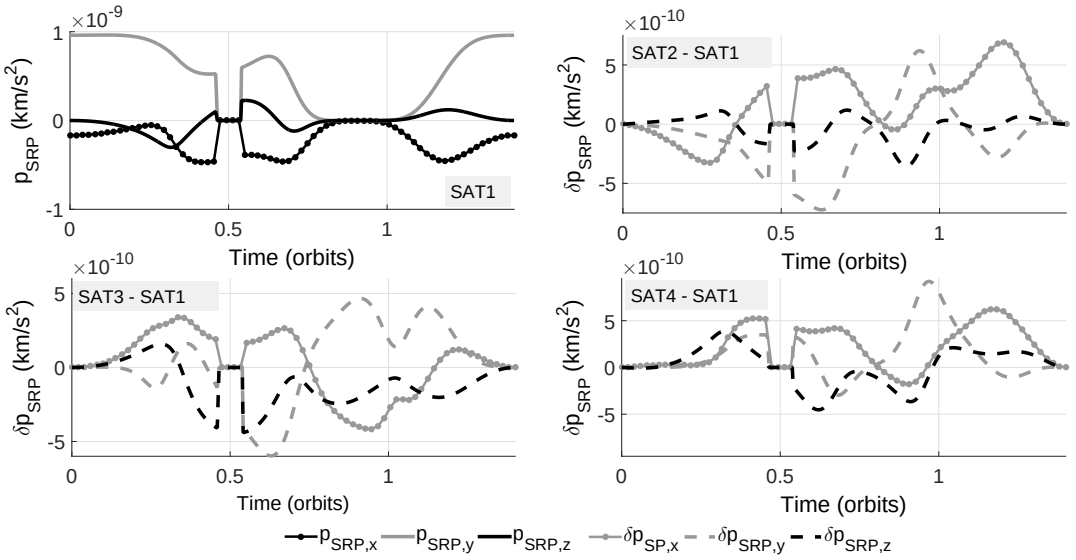


Fig. 8.19: SRP perturbation history for the PCF to GCF case in MEO with Ω_1.

the fact that when the line of nodes is aligned with the Earth-Sun direction, the SRP goes to zero when the satellites are in the Earth shadow, as can be seen in Fig. 8.19. From this figure, moreover, the magnitude of the SRP can be detected and compared to the drag intensity reported in Fig. 8.15. Even though the SRP magnitude is less than the one provided by the drag the maneuver can be accomplished since the gravitational field intensity in MEO is less than in LEO.

The out-of-plane effect of the SRP on the reconfiguration maneuver can be seen in Fig. 8.20, where all the six parameters are modified. It is noteworthy that the variations in δi_x and δi_y are no more periodical as they were in Fig. 8.16 where their histories where induced by the conservative perturbations only. However, also in this case δa and $\delta\lambda$ converges to a zero final value, which is consistent with the expected results.

Finally, the same characteristics described in LEO with regards to the rotation angles and the exposed area are detected in MEO. Also in this case the rotation angles histories are such that the exposed area tends to have minimum or maximal values, thus representing a nice approximation of a bang-bang solution. For brevity, the torques required for the attitude maneuvers are not reported. However, the necessary torques in MEO are always lower than the ones obtained in LEO since the forces ruling the spacecrafts dynamics are very weak, the maneuver time is longer than in LEO and no fast attitude maneuvers are required.

For investigation purposes, a test has been run using a chief orbit with $i = 90$ degree and Ω_2. In this case, the in-plane component of the SRP is so small that the optimizer cannot find a feasible reconfiguration maneuver.

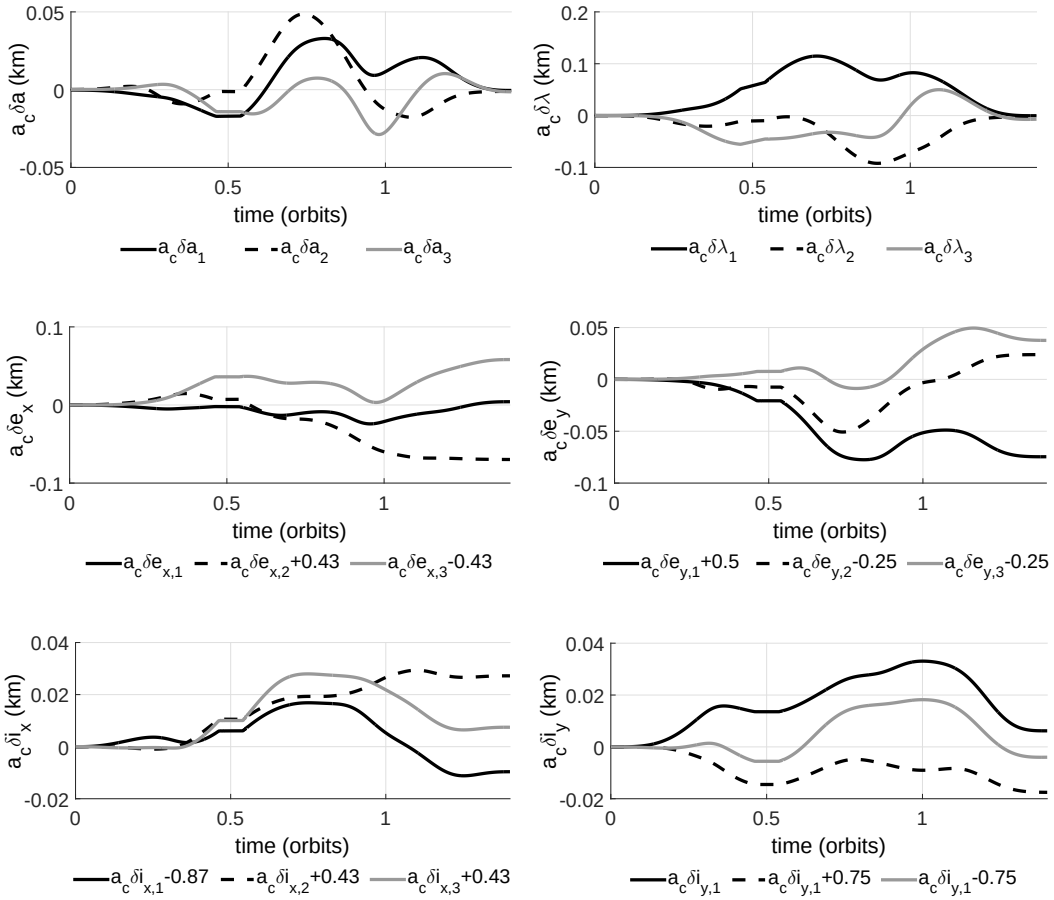


Fig. 8.20: ROE history for the PCF to GCF maneuver, in MEO with $\boldsymbol{\Omega_1}$ (ROE components have been translated to zoom all the deputies' behavior).

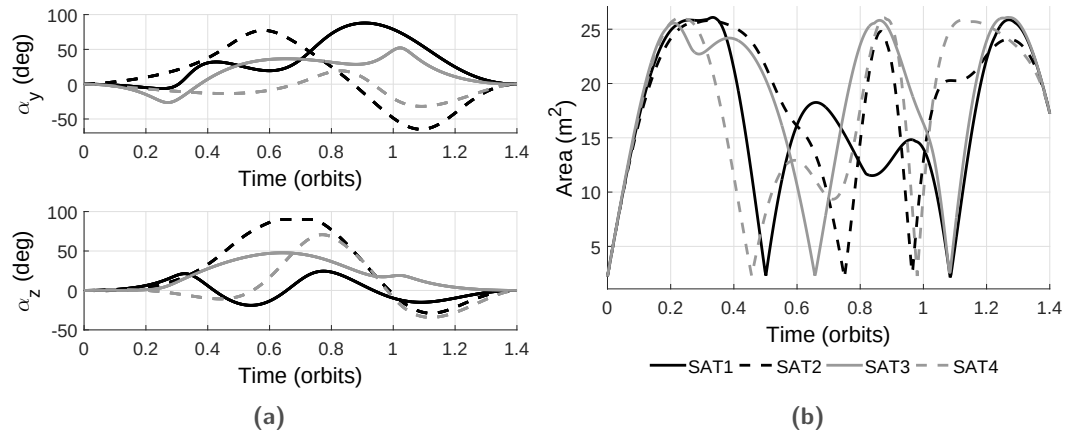


Fig. 8.21: Rotation angles (a) and exposed areas (b) for PCF to GCF maneuver, in MEO with Ω_1.

8.6.4 Case 3: PCF-GCF reconfiguration in GEO

The chief orbital parameters are reported in Table 8.9, whereas the performances obtained in GEO are reported in Table 8.10. In this case, a GCF to GCF maneuver imposing an increased final radius of 1.2 km has been performed. For all the reported cases, approximately one orbit is sufficient for the maneuver. It is noteworthy that, as in the MEO case, the final radius of the two trajectories after the maneuver is slightly different from the expected one due to the presence of in-plane and out-of-plane components.

Detailed results and figures are not reported for the PCF to GCF maneuvers as they are quite similiar to those already reported for the cases in LEO and MEO. However, the GCF to GCF maneuver is reported through Fig. 8.22 and Fig. 8.23. As described in Sec. 8.4.4, this maneuver can be accomplished since the SRP perturbation has an out-of-plane component that can be used to increase the radius

Tab. 8.9: Chief orbital parameters in GEO.

Parameter	Value	Parameter	Value
a (km)	42168	ω (deg)	0
e	0	i (deg)	1
Ω (deg)	10	M (deg)	0

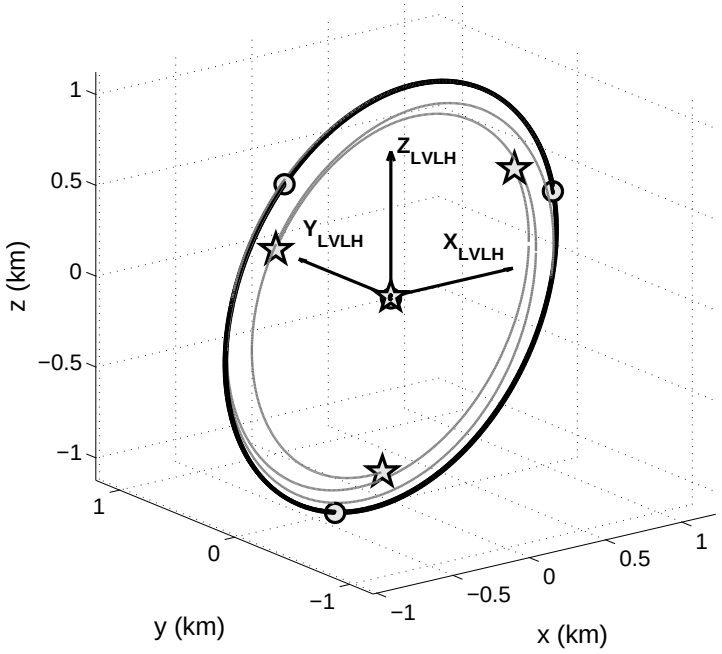


Fig. 8.22: GCF to GCF maneuver in the $\mathscr{L}$ Reference System, in GEO with $R_f^* = 2$.

Tab. 8.10: Maneuver performances achievable with SRP in GEO.

	$K_{zx}(t_0)$	$R(t_0)$	$K_{zx}(t_f)$			t_f	$\frac{R(t_f)}{R(t_0)}$
	(-)	(km)		(-)		(orbits)	(-)
	all deputies		SAT2	SAT3	SAT4	all deputies	
			PCF-GCF maneuvers				
PCF to GCF	2.000	1.000	1.728	1.729	1.716	0.960	1.139
GCF to PCF	1.732	1.000	1.963	1.979	1.984	0.936	0.862
			GCF to GCF maneuver imposing $R_f^* = 1.2$ km				
GCF to GCF	1.732	1.000	1.712	1.714	1.710	1.048	1.194

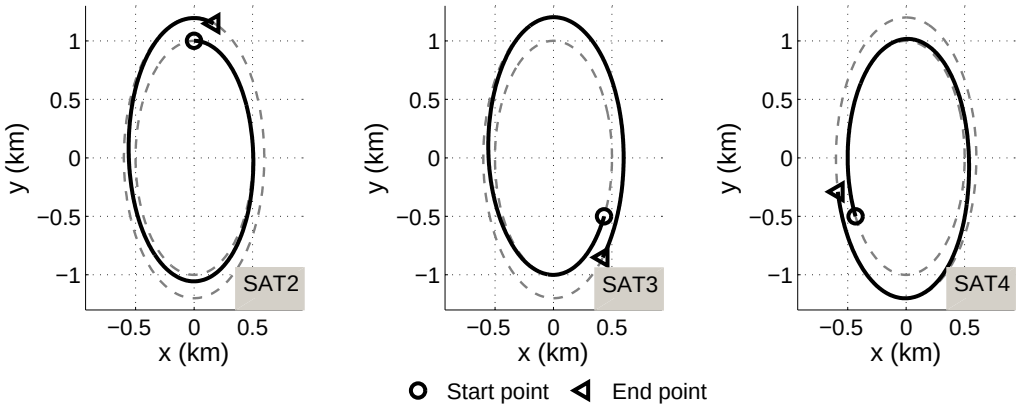


Fig. 8.23: GCF to GCF maneuver in the $\mathscr{L}$ Reference System, xy projection, in GEO with $R_f^* = 2$.

of the relative trajectory. By means of the reported figures, the reader can appreciate the performances achieved with the proposed maneuver. The satisfaction of the end-point constraints is achieved and errors on the values of K_{zx} and R_f are small. Such errors might be due to the IPSO difficulty in reaching the optimal maneuver or might be due to the simultaneous effects of the in-plane and out-of-plane components of the SRP. Nonetheless, errors are so small that in the worst case only a small amount of external control would be needed to achieve the final desired state.

8.6.5 Case 4: ATF reconfiguration

In this case the maneuver consists in changing the inter-distance between successive ATF satellites starting from an initial value of $\partial = 1$ km and arriving at a final value of $\partial = 1.5$ km. The initial conditions of the simulations have been reported in Table 8.11. At the beginning, the satellite are disposed along the $y_{\mathscr{L}}$ axis according to the vector $[-1.5, -0.5, 0.5, 1.5]$. At the end of the maneuver, the

Tab. 8.11: Initial conditions for the ATF reconfiguration.

	$\boldsymbol{r}_{\mathscr{L}}(t_0)$			$\dot{\boldsymbol{r}}_{\mathscr{L}}(t_0)$		
	x [km]	y [km]	z [km]	$\dot{x}$ [km/s]	$\dot{y}$ [km/s]	$\dot{z}$ [km/s]
SAT 1	0	- 1.5	0	0	0	0
SAT 2	0	- 0.5	0	0	0	0
SAT 3	0	+0.5	0	0	0	0
SAT 4	0	+1.5	0	0	0	0

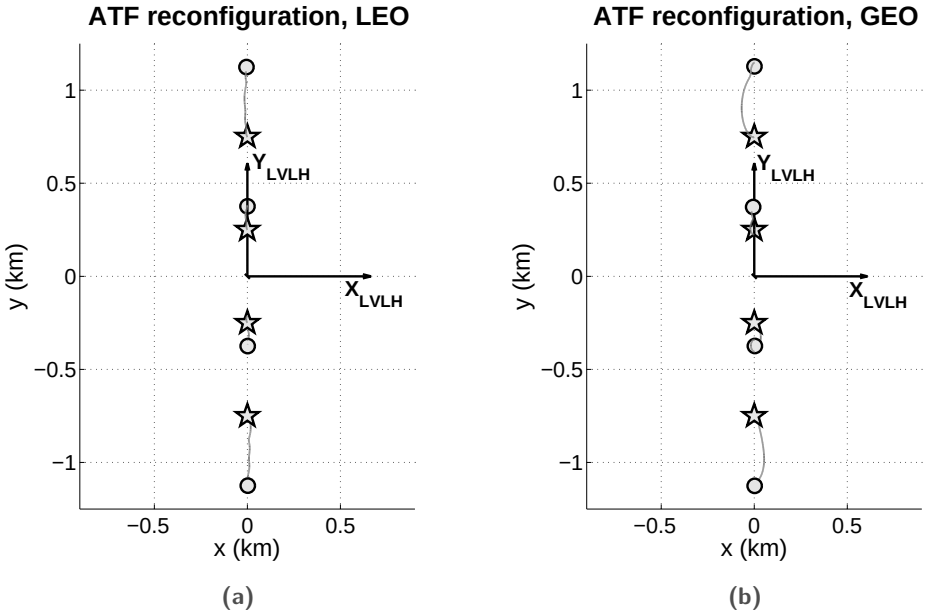


Fig. 8.24: ATF reconfiguration maneuver in the LVLH Coordinate System $\mathscr{L}$, in LEO and GEO.

satellites must be along the $y_{\mathscr{L}}$ axis according to the vector $[-2.25, -0.75, 0.75, 2.25]$ with zero relative velocity. The geometrical center of the formation is empty, but we can associate to this position a reference orbit described by the initial conditions reported in Table 8.9. Results are reported in Table 8.12, where the position and velocity errors are the average errors evaluated over one orbit propagation after the end of the maneuver. In contrast to the previous test case, this maneuver may be accomplished for various altitudes taken into account with satisfactory precisions. In fact, the relative error with respect to the final imposed position is always less than 1% and the final velocities are below 1 cm/s. Clearly, the velocity errors are smaller in magnitude for higher altitude trajectories as a result of the orbital angular velocities of MEO and GEO being lower than the LEO angular velocities. However, the precision of the maneuver is quite the same comparing the different altitudes.

It is worth noting that the drag and the solar radiation pressure perturbations lead to different reconfiguration trajectories. In Fig. 8.24(a) the LEO case at 400

Tab. 8.12: ATF reconfiguration (inter-distance from 1 km to 1.5 km).

Reference Orbit		Position error (m)	Relative position error (·)	Velocity error (cm/s)	Maneuver Time (orbits)
LEO $H = 400$ km	SAT1	2.38	0.21%	0.40	4.81
	SAT2	1.81	0.48%	0.31	
	SAT3	1.07	0.29%	0.14	
	SAT4	5.05	0.45%	0.84	
LEO $H = 600$ km	SAT1	1.54	0.14%	0.24	9.71
	SAT2	2.22	0.59%	0.40	
	SAT3	0.73	0.20%	0.11	
	SAT4	3.04	0.27%	0.49	
LEO $H = 800$ km	SAT1	0.90	0.21%	0.11	13.77
	SAT2	1.60	0.43%	0.24	
	SAT3	0.32	0.09%	0.04	
	SAT4	2.60	0.23%	0.40	
MEO $H = 20000$ km	SAT1	4.04	0.36%	0.09	1.58
	SAT2	3.51	0.94%	0.10	
	SAT3	2.67	0.71%	0.04	
	SAT4	7.04	0.63%	0.15	
GEO $H = 35800$ km	SAT1	2.13	0.19%	0.02	1.87
	SAT2	3.14	0.84%	0.03	
	SAT3	3.42	0.91%	0.07	
	SAT4	2.79	0.25%	0.01	

km is reported, while in Fig. 8.24(b) the GEO case is shown. As it can be seen, the trajectories are quite different: with the drag, the re-orientation maneuver is closer to the $y_{\mathscr{L}}$ axis than in the case when using the solar radiation pressure perturbation.

8.7 Endnotes

In this chapter a novel approach has been proposed for the planning of attitude maneuver allowing the reconfiguration of satellite formation by mean of the drag/solar radiation forces. The planner takes into account all the most dominant perturbation forces which can affect the maneuver. It has been shown that the intensity and/or the direction of the drag/solar pressure perturbation forces may be changed with simple attitude maneuvers, thus leading to a feasible reconfiguration satisfying all the imposed formation constraints.

A major result is reported, that is we are able to obtain any orientation of the relative motion plane by modifying the intensity of the drag and the solar radiation pressure effects with attitude maneuvers. The results obtained computing the maneuver from the projected circular formation to the general circular formation, and vice-versa, confirm this finding.

In low Earth orbit, the drag can only affect the in-plane Relative Orbital Elements, making them vary accordingly to the expected results and thus verifying the validity of the model. With added improvements to the model, as well as refinements to the planning algorithm based on the inverse dynamics particle swarm optimization, we achieve closer to theoretical time optimal bang-bang solutions. The presence of the gravity gradient torque does not affect the possibility to perform the maneuvers.The passive reconfiguration using perturbing forces in low Earth and geostationary orbits does not allow one to increase the major and minor axes of the formation ellipse.

The differences in the nature of the reconfiguration maneuvers using drag for low altitude orbits and solar radiation pressure for high altitude orbits is charactezied. All the Relative Orbit Elements vary in the presence of the solar radiation pressure as a consequence of the maneuver.

The initial and final attitude of all the satellites are imposed a-priori using an inverse dynamic approach with respect to the attitude dynamics. The results are satisfactory for all the considered cases: circular formations may be reconfigured in about one orbital period and along track formations are reconfigured with final position and velocity errors less than 10 m and 1 cm/s, respectively. The planner gives the possibility to underline the feasibility of the maneuver and to find out a near time-optimal guide.

The reported results demonstrate the feasibility of the proposed approach and suggest further research to obtain the optimal design of missions utilizing the advantages from perturbation-based reconfiguration maneuvers. In fact, given the great precision that can be achieved, the proposed maneuvers may be used both as a default reconfiguration strategy and as a backup system in the case of a failure in another control subsystem.

9

Near-Optimal Maneuvers for Satellite Formations with Inverse Dynamics Approach and Differential Evolution

Abstract

This chapter presents an open-loop planner for near time-optimal maneuvers performed by satellite formations during proximity operations. The optimal control technique is based on the inverse dynamics approach and the Differential Evolution. The linearized dynamical model including the J_2 perturbation is taken into account. The Differential Evolution algorithm with Local Neighborhood is employed, where the risk to stop at local minima is reduced. The solution is computationally efficient but is sub-optimal since the polynomial approximation of the kinematics cannot guarantee a bang-bang control policy. Results are reported to evaluate the performances of the technique and a Monte Carlo simulation has been run to prove the efficiency of the planner over a variety of different scenarios. Moreover, the Chebyshev and the B-spline approximations are compared to establish that the latter approach guarantees better results than the former in terms of maneuver time and computational effort.

Nomenclature

$\boldsymbol{r}$	=	Inertial position (km)	$\boldsymbol{\rho}$	=	Relative position vector (km)
$\boldsymbol{u}$	=	External acceleration (km/s^2)	$\boldsymbol{f}$	=	Perturbation acceleration (km/s^2)
μ	=	Gravitational constant (km^3/s^2)	ω	=	Angular velocity (rad/s)
$\bar{m}$, $\bar{n}$	=	Gravitational coefficients	J_2	=	Zonal gravitational harmonic
F_l, F_g	=	DE scale factor	$(\cdot)^N$	=	Numerical approximation
w	=	Differential Evolution weight	λ	=	DE scale factor
$N_{\mathcal{P}}$	=	Approximation parameters number	CR	=	DE cross-over rate
j, G, i	=	Indices	$\boldsymbol{X}$	=	DE individual
J	=	Performance index	$\boldsymbol{L}$	=	Local Mutation Vector
$\boldsymbol{G}$	=	Global Mutation Vector	$\boldsymbol{V}$	=	DE Mutant Vector
$\boldsymbol{U}$	=	DE trial vector	$\mathscr{I}$	=	ECI coordinate system
$\mathscr{L}$	=	LVLH coordinate system	$\boldsymbol{\psi}$	=	Covector
$(\cdot)_{(c,d)}$	=	Chief/deputy quantity	$(\cdot)_{(0/f)}$	=	Initial/final time value
$(\cdot)^*$	=	Optimal quantity	$\boldsymbol{x}$	=	State

$\mathcal{H}$	=	Pontryagin Hamiltonian function	$\mathcal{C}$	=	Chebyshev approximation
T_k	=	Chebyshev polynomial	$\mathcal{B}$	=	B-spline approximation
$\mathcal{N}_{i,j}$	=	B-spline basis functions	$\boldsymbol{\mathcal{K}}$	=	B-spline knot vector
τ	=	Independent variable	$\boldsymbol{a}, \boldsymbol{b}$	=	Approximation coefficients vectors
N_{viol}	=	Number of violations	N_{max}	=	DE maximum iterations
$K_{(\cdot)}$	=	Dimensional coefficients	u_{max}	=	Maximum available control
R_L	=	Local neighborhood radius	$\mathcal{S}$	=	Size of the DE population

9.1 Introduction

To analyze the SFF dynamics several analytical models have been proposed, from the easiest set of equations given by the Hill-Clohessy-Whiltshire (HCW) model [131], to very accurate models including orbital perturbation or nonzero eccentricities [132]. Moreover, different sets of elements may be used to parametrize the equations, the most important being Cartesian coordinates, as in [131], differential orbital elements [132] and relative orbital elements [158]. In this chapter the linear relative motion model developed by Schweighart and Sedwick (SS) and including the J_2 perturbation is employed [168, 169].

This chapter deals with planning time-optimal maneuvers for proximity operations. The optimal control theory may be applied to the HCW and the SS equations. In both cases, the Pontryagin Maximum Principle described in Sec. 1.4.2 can give some information about the *form* of the optimal control, but the analytical solution for the optimal control and trajectory histories cannot be easily obtained. When one of the planning goal is the computational efficiency of the algorithms, sub-optimal, feasible trajectories may be accepted as solutions of guidance problems. Even though the computational complexity of an optimal problems is primarily associated with the problem parametrization (see for instance the analysis in Ref. [121] with respect to pseudospectral approaches), heuristics algorithms have been often employed to reach solutions offering nice compromises between optimality and computational cost.

Several works may be found in literature where heuristic approaches have been employed for planning trajectories in the frame of relative motion operations. For instance, the PSO has been used in [8, 170], the Evolutionary Computation (EC) has been exploited in [171] and a Genetic Algorithm (GA) has been used in [172, 173].

In this application, the Differential Evolution (DE) algorithm [174] is used and the inverse dynamics approach for differentially flat systems is employed. As a consequence, the object of the optimization will be the parameters describing the trajectories of the satellites, while velocities, accelerations and control histories will

be obtained with easy derivative operations using the kinematic and dynamical constraints of the reference equations of motion.

In the previous chapters of this thesis, the PSO technique has been applied to solve the NLP problem. Based on some recent analysis and comparisons between DE and PSO reported in literature [175, 176, 177], DE seems to outperform PSO. As a consequence, DE will be employed in this work, and a comparison analysis with PSO will be performed to validate the results. The DE algorithm has many features in common with the Genetic Algorithms (GAs). However, while GAs solutions are generally coded using the binary alphabet (for instance, see [172]) and must be carefully handled to deal with real-valued optimization parameters [178], DE is explicitly designed to solve optimization problems in the set of the real number [179] as well as PSO. It is noteworthy that the comparison between DE and PSO is performed considering a particular test case, as a general assessment stating that the DE can *always* outperform the PSO would contradict the No free lunch theorem in Ref. [180].

With regard to previous studies concerning minimum-time maneuvers for proximity operations, some results may be found in [181, 182, 183]. Often, minimum-time maneuvers are searched for when dealing with low thrust maneuvers, as in the cases where the differential drag force is involved [8, 184, 185].

Another goal of this investigation is to accomplish a comparison analysis between two global approximation approaches, the first based on Chebyshev polynomails and the second based on B-splines curves. Methods based on Chebyshev polynomials [186, 187] represent an effective approach to find near *minimax* approximation [188, 189], and for this reason they are widely used in many numerical analysis. The strongest advantage of Chebyshev polynomials is that the norm of the basis functions for the interpolation is always less or equal to one, thus giving a smooth and well-behaved function. On the other hand, B-spline curves often give very good results using minimal number of support functions, as already described in Sec. 4.2. The comparison with Chebyshev polynomials is performed to assess what interpolation method is more useful for the examined test cases.

This chapter is organized as follows. Sec. 9.2 describes the dynamical model considered for the optimization problem and presents some results obtained from the application of the optimal control theory to the proposed problem. In Sec. 9.3 the inverse dynamics approach is described, whereas Sec. 9.4 gives the fundamental details to understand the characteristics of the DE algorithm. Sec. 9.5 deals with the description of the details concerning the implementation of the proposed optimizer. Numerical results and discussions related to the efficiency of the proposed technique are reported in Sec. 9.6. Finally, concluding remarks are given in Section 9.7.

9.2 Optimal control problem statement

Let us recall some preliminary concepts concerning spacecraft formation flying which have been already introduced in Chapter 8. Accordingly, with reference to Fig. 8.1, let us suppose to have a formation with two satellites in close relative motion, the *chief* (c) and the *deputy* (d). In the Earth Centered Inertial (ECI) coordinate system $\mathcal{I} = \{X_\mathcal{I}, Y_\mathcal{I}, Z_\mathcal{I}\}$, the satellite dynamics in the presence of an external acceleration $\boldsymbol{u}$, provided by engines, and a perturbation acceleration $\boldsymbol{f}$, depending on environmental forces, is

$$\ddot{\boldsymbol{r}} = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{u} + \boldsymbol{f}\,, \tag{9.1}$$

where $\mu = 3.986 \cdot 10^5$ km^3/s^2 is the Earth gravitational constant, $\boldsymbol{r}$ is the position vector of the satellite and r its magnitude. Let $\boldsymbol{\rho}_d = \boldsymbol{r}_d - \boldsymbol{r}_c = [x_d,\, y_d,\, z_d]^T$ be the distance between the chief and the deputy. The relative motion equations are referred to the rotating Local-Vertical, Local-Horizontal (LVLH) reference frame $\mathcal{L} = \{X_\mathcal{L}, Y_\mathcal{L}, Z_\mathcal{L}\}$ shown in Fig. 8.1, where $X_\mathscr{L}$ points from the center of the Earth to the origin of $\mathcal{L}$, $Z_\mathcal{L}$ is perpendicular to the orbital plane and $Y_\mathcal{L}$ completes a right-hand Cartesian coordinate system.

Considering the chief satellite on a circular orbit (which implies $\omega = \sqrt{\mu/r_c^3}$), including the J_2 gravitational effect in $\boldsymbol{f}$ and imposing $\boldsymbol{u}_c = \boldsymbol{0}$, the relative motion is described by the Schweighart and Sedwick (SS) model [168, 169],

$$\begin{aligned} \ddot{x}_d - 2\bar{m}\dot{y}_d - (4\bar{m}^2 - \bar{n}^2)x_d &= u_x\,, \\ \ddot{y}_d + 2\bar{m}\dot{x}_d &= u_y\,, \\ \ddot{z}_d + (2\bar{m}^2 - \bar{n}^2)z_d &= u_z\,, \end{aligned} \tag{9.2}$$

where $\bar{m} = \omega\sqrt{1 + k_{J_2}}$, $\bar{n} = \omega\sqrt{1 - k_{J_2}}$ and $k_{J_2} = \dfrac{3J_2R_e^2}{8r_c^2}(1 + 3\cos(2i_c))$.

Let $\boldsymbol{x} = [x_1,\, x_2,\, x_3,\, x_4,\, x_5,\, x_6]^T = [\boldsymbol{\rho}_x^T,\, \dot{\boldsymbol{\rho}}_x^T]^T \in \mathbb{R}^{N_x}$ be the state and $\boldsymbol{u} = [u_x, u_y, u_z]^T \in \mathbb{R}^{N_u}$ be the control. Clearly, $N_x = 6$ and $N_u = 3$. Eq. (9.2) may be written in state-space form as

$$\dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}\boldsymbol{u} \tag{9.3}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are

$$\mathbf{A} = \left[\begin{array}{ccc|ccc} \multicolumn{3}{c|}{\mathbf{0}_{3\times3}} & \multicolumn{3}{c}{\mathbf{I}_{3\times3}} \\ \hline \alpha & 0 & 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & -\beta & 0 & 0 \\ 0 & 0 & -\gamma & 0 & 0 & 0 \end{array}\right], \; \mathbf{B} = \left[\begin{array}{ccc} \multicolumn{3}{c}{\mathbf{0}_{3\times3}} \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right], \tag{9.4}$$

and the constants α, β and γ may be easily obtained from Eq. (9.2). Let $\boldsymbol{\rho}_0$ and $\dot{\boldsymbol{\rho}}_0$ be the deputy relative position and velocity at $t = t_0 = 0$ and $\boldsymbol{\rho}_f$ and $\dot{\boldsymbol{\rho}}_f$ be the desired final relative state $t = t_f$. Moreover, the reconfiguration maneuver must be accomplished in the shortest final time t_f^*. The time-optimal problem considered in this work can be summarized as follows:

$$\begin{aligned} &\text{Find } \{\boldsymbol{x}, \boldsymbol{u}, t_f\}^* \\ &\text{minimizing } \; J = t_f - t_0 \\ &\text{subject to, } \; \forall t \in [t_0, t_f], \\ \text{dynamic constraints}: \quad & \dot{\boldsymbol{x}} = \mathbf{A}\boldsymbol{x} + \mathbf{B}\boldsymbol{u}, \\ \text{boundary conditions}: \quad & \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \; \boldsymbol{x}(t_f) = \boldsymbol{x}_f, \\ \text{control constraint}: \quad & \|\boldsymbol{u}(t)\|_\infty - u_{max} \leq 0. \end{aligned} \tag{9.5}$$

Following the notation in [29], a necessary condition for the optimality of $\boldsymbol{u}^*(t)$ is provided by the Pontryagin's Maximum Principle (PMP). Therefore, let the Pontryagin's Hamiltonian be

$$\mathcal{H}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\psi}(t), t) = \boldsymbol{\psi}^T(t)[\mathbf{A}\boldsymbol{x}(t) + \mathbf{B}\boldsymbol{u}(t)], \tag{9.6}$$

where $\boldsymbol{\psi}(t) = [\boldsymbol{\psi}_\rho, \boldsymbol{\psi}_{\dot{\rho}}] \in \mathbb{R}^{N_x}$ is the costate vector ($\boldsymbol{\psi}_\rho$ and $\boldsymbol{\psi}_{\dot{\rho}}$ have the same dimensions of $\boldsymbol{\rho}$ and $\dot{\boldsymbol{\rho}}$, respectively). According to PMP and normalizing the control so that $\|\boldsymbol{u}\|_\infty \in [0, 1]$ (which means that in order to normalize the control components dividing by u_{max}), the components of $\boldsymbol{u}^*$ are found to be

$$\boldsymbol{u}^* = \text{sign}(\boldsymbol{\psi}_{\dot{\rho}}^*). \tag{9.7}$$

Eq. (9.7) tells us that the optimal external input for the SS dynamical models is a bang-bang control policy. However, even though the evolution of the costate is known to be

$$\boldsymbol{\psi}(t) = \Phi^T(t_0, t)\boldsymbol{\psi}(t_0), \tag{9.8}$$

where $\Phi(t, t_0)$ is the state transition matrix for $\boldsymbol{A}$ (see [25] or Appendix C in [190]), there is no analytical way to determine the costate initial conditions $\boldsymbol{\psi}(t_0)$ necessary to solve for Eq. (9.8) and properly defining $\boldsymbol{\psi}_{\dot{\rho}}$ in Eq. (9.7). Therefore, it is necessary to proceed with numerical methods to evaluate $\boldsymbol{u}^*$. In this work, near-optimal

numerical solutions are provided using a DF formulation of the proposed OCP. With respect to other approaches, the proposed method relies on the minimum number of optimization parameters thus reducing the required computational effort.

9.3 Inverse-dynamics parametrization

The numerical solution of an optimal control problem is based on the assumption that the original problem, defined over some specified function set (e.g., the continuous and differentiable functions set for the state, or the Lebesgue integrable functions for the control), may be converted into a non-linear programming (NLP) problem involving a finite number of parameters [11, 45]. An inverse-dynamics OCP formulation can be employed if the external control can be written as a function of the state. In this case, the state $\boldsymbol{x}$ is expressed with a polynomial approximation $\boldsymbol{x}^N$ and a NLP technique searches for the optimal values of the $N_{\mathcal{P}}$ coefficients of each state component (for a total of $N_x N_{\mathcal{P}}$ optimization parameters). On the contrary, standard direct-dynamics approaches (e.g, [100]) are based on the integration of an approximated control $\boldsymbol{u}^N$ and the NLP technique searches for the optimal values of the $N_{\mathcal{P}}$ coefficients of each control component (for a total of $N_u N_{\mathcal{P}}$ optimization parameters). Collocation and pseudospectral methods approximate both state and control, having a total number of $(N_x + N_u)N_{\mathcal{P}}$ optimization parameters. The inverse dynamics approach overcomes some direct dynamics issues such as low computational speeds and integration numerical errors and has a reduced number of optimization parameters with respect to collocation and pseudospectral methods.

In this paper, the DF formulation is employed, which can be seen as a sub-class of the inverse-dynamics approach. In cases where $N_x > N_u$ (as the one introduced in Sec. 9.2), the DF formulation is based on the so-called flat output $\boldsymbol{y}(t)$ with the same dimension of the control $\boldsymbol{u}(t)$ (see Ref. [32, 191]). Accordingly, the number of optimization parameters in the transcribed NLP problem is $N_u N_{\mathcal{P}}$. Let $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t))$ be an autonomous dynamical system and $\boldsymbol{y}(t) \in \mathbb{R}^{N_u}$ be the flat output. If such a flat output exists, then state and control can be written as

$$\boldsymbol{x} = \boldsymbol{a}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, ..., \boldsymbol{y}^{(\beta)}\right) \ , \quad \boldsymbol{u} = \boldsymbol{b}\left(\boldsymbol{y}, \dot{\boldsymbol{y}}, ..., \boldsymbol{y}^{(\beta+1)}\right), \tag{9.9}$$

where the value of $\beta \in \mathbb{N}$ depends on the problem. For the problem outlined in Sec. 9.2, let $\boldsymbol{y} = \boldsymbol{\rho}$. Hence, $\beta = 1$ and the full state $\boldsymbol{x}$ is obtained via time-differentiation of $\boldsymbol{\rho}$ whereas $\boldsymbol{u} = \boldsymbol{b}\,(\boldsymbol{\rho},\ \dot{\boldsymbol{\rho}},\ \ddot{\boldsymbol{\rho}}\ )$ as can be seen from Eqs. (9.2),(9.3),(9.4). The problem of Sec. 9.2 may be rewritten as a function of the flat output as

$$
\begin{aligned}
&\text{Find } \{\boldsymbol{\rho}, t_f\}^* \\
&\text{minimizing } \; J = t_f - t_0 \\
&\text{subject to, } \; \forall t \in [t_0, t_f], \\
&\text{boundary conditions}: \quad \boldsymbol{\rho}(t_0) = \boldsymbol{\rho}_0, \; \boldsymbol{\rho}(t_f) = \boldsymbol{\rho}_f, \\
&\qquad\qquad\qquad\qquad \dot{\boldsymbol{\rho}}(t_0) = \dot{\boldsymbol{\rho}}_0, \; \dot{\boldsymbol{\rho}}(t_f) = \dot{\boldsymbol{\rho}}_f, \\
&\text{control constraint}: \quad \| \boldsymbol{b}\left(\boldsymbol{\rho}, \; \dot{\boldsymbol{\rho}}, \; \ddot{\boldsymbol{\rho}}\right) \|_\infty - u_{max} \leq 0.
\end{aligned}
\tag{9.10}
$$

As can be seen, the dynamics constraint is no longer reported as it is a-priori satisfied with the proposed problem formulation. To further simplify the problem, the flat output approximation can be chosen to satisfy boundary constraints a-priori. This is an easy operation as the boundary constraints satisfaction is related to a clever choice of the approximating polynomial coefficients (see Sec. 9.5 for details). Defining $\mathcal{D}_B$ as the set of approximating function $\boldsymbol{x}^N$ satisfying the boundary constraints, the OCP formulation is

$$
\begin{aligned}
&\text{Find } \{\boldsymbol{\rho}^N, t_f\}^* \text{ such that } \boldsymbol{\rho}^N \in \mathcal{D}_B \\
&\text{minimizing } \; J = t_f - t_0 \\
&\text{subject to, } \; \forall t \in [t_0, t_f], \\
&\text{control constraint}: \quad \| \boldsymbol{b}\left(\boldsymbol{\rho}^N, \; \dot{\boldsymbol{\rho}}^N, \; \ddot{\boldsymbol{\rho}}^N\right) \|_\infty - u_{max} \leq 0.
\end{aligned}
\tag{9.11}
$$

Hence, the advantages of the DF formulation are:

1. The minimum number of optimization parameters is employed.

2. The control policy is obtained in an analytical closed form avoiding the integration of dynamics equations.

3. The initial and final conditions are automatically respected since they are imposed a priori in the polynomial approximation of $\boldsymbol{\rho}$.

9.4 Differential evolution algorithm

In this paper the Differential Evolution (DE) has been implemented taking advantage of different features reported in literature. As in the author's previous works dealing with PSO, the DE algorithm has been implemented in order to find a compromise between exploitation and exploration of the search space. The DE is a stochastic, population-based algorithm introduced in 1995 by K. Price and R.

Storn [174] to solve optimization problems over continuous spaces. The size of the population $\mathcal{S}$ is fixed and the members of the population are candidate solutions called individuals, represented by vectors $\boldsymbol{X}_{i,G}$ which components are the decision variables of the problem,

$$\boldsymbol{X}_{i,G} = [x^1_{i,G}\, x^2_{i,G} \dots\, x^m_{i,G}]\,. \tag{9.12}$$

In Eq. (9.12), the i^{th} individual at the generation G is considered, while m is the number of decision variables. At the beginning of the DE algorithm ($G = 0$), components of $\boldsymbol{X}_{i,0}$ are randomly generated within a user-defined interval with lower and upper bound given as $\boldsymbol{x^L}$ and $\boldsymbol{x^H}$, i.e.

$$x^L \leq x^j_{i,0} \leq x^H,\ i \in [1, ..., \mathcal{S}], j \in [1, ..., m]. \tag{9.13}$$

Each individual is associated to a scalar index value $J_{i,G}$ evaluated through the performance index J which takes into account the goal of the optimization and the constraints (see Sec. 9.5.3). The algorithm is aimed at making the individuals evolve towards the global optimum corresponding to the minimum of the performance index. The optimization algorithm is based on a continuous perturbation of the population: old individuals are substituted by new trial individuals generated through the mutation and cross-over operators to explore new regions of the Feasible Search Space (FSS) which is the set of all the feasible solutions.

Several strategies for mutation, cross-over and selection have been produced [179], as in [192] where variable coefficients for the mutation vector have been presented. In this paper, to reduce the risk to stop at local minima, DE with Local Neighborhood is employed [193]. This technique resembles the PSO local strategy which has been used in [1, 2, 122]. The algorithm initialization is based on a uniform random distribution of the individuals within the FSS. For each *target* vector $\boldsymbol{X}_{i,G}$, a *mutant vector* $\boldsymbol{V}_{i,G}$ is produced as a linear combination of two inputs, $\boldsymbol{L}_{i,G}$ and $\boldsymbol{G}_{i,G}$.

For each member of the population, a neighborhood of radius R_L is defined as $N_i = \{\boldsymbol{X}_{j,G} \mid i - R_L \leq j \leq i + R_L\}$. Then, for each vector $\boldsymbol{X}_{i,G}$ a local mutation operator is applied, in order to obtain a *Local Mutation vector* $\boldsymbol{L}_{i,G}$

$$\boldsymbol{L}_{i,G} = \boldsymbol{X}_{i,G} + \lambda(\boldsymbol{X}_{l,G} - \boldsymbol{X}_{i,G}) + F_l(\boldsymbol{X}_{r_1,G} - \boldsymbol{X}_{r_2,G})\,, \tag{9.14}$$

where λ and F_l are the scale factors for the local mutation, $\boldsymbol{X}_{l,G}$ is the individual with the best performance index in N_i, r_1 and r_2 are random integers different from

i such that $\boldsymbol{X}_{r_1,G}$ and $\boldsymbol{X}_{r_2,G}$ are in N_i. Similarly, the *Global Mutation vector* $\boldsymbol{G}_{i,G}$ is created as

$$\boldsymbol{G}_{i,G} = \boldsymbol{X}_{i,G} + \lambda(\boldsymbol{X}_{g,G} - \boldsymbol{X}_{i,G}) + F_g(\boldsymbol{X}_{r_3,G} - \boldsymbol{X}_{r_4,G})\,, \tag{9.15}$$

where F_g is the scale factor for the global mutation, $\boldsymbol{X}_{g,G}$ is the individual with the best performance index in the entire population, $\boldsymbol{X}_{r_3,G}$ and $\boldsymbol{X}_{r_4,G}$ are chosen randomly in the entire population at the step G such that r_3 and r_4 are different from i. In this work, the coefficient λ is not fixed, but it linearly decreases during the search of the optimal solution, by exploiting the idea introduced in [192]. In this way, the DE exploitation ability inside the final convergence sub-set of the search space is enhanced. Finally, the mutant vector $\boldsymbol{V}_{i,G}$, is evaluated combining $\boldsymbol{L}_{i,G}$ and $\boldsymbol{G}_{i,G}$ as

$$\boldsymbol{V}_{i,G} = w\boldsymbol{G}_{i,G} + (1-w)\boldsymbol{L}_{i,G}\,. \tag{9.16}$$

To emphasize the DE exploration ability at the beginning of the optimization process and improve the exploitation characteristic when the population is close to converge, the parameter w combining local and global mutation linearly increases during the evolution as

$$w = w_{min} + (w_{max} - w_{min})\frac{G}{G_{max}}. \tag{9.17}$$

In Eq. (9.17), G_{max} is the number of user-defined steps to pass from the minimum value, w_{min}, to the maximum one, w_{max}. After the mutation, the trial vectors $\boldsymbol{U}_{i,G} = [u^1_{i,G}\, u^2_{i,G} \ldots u^m_{i,G}]$ are generated by applying the cross-over operator between the target vector $\boldsymbol{X}_{i,G}$ and the corresponding mutant vector $\boldsymbol{V}_{i,G}$. The binomial cross-over is expressed as

$$u^j_{i,G} = \begin{cases} v^j_{i,G}, & \text{if } R_j \leq CR \text{ or } j = j_{rand}\,, \\ x^j_{i,G}, & \text{otherwise}\,, \end{cases} \tag{9.18}$$

where, R_j is a randomly picked number within the interval $(0,1)$, the Cross-over Rate $CR \in [0,1)$ is a user-defined constant and j_{rand} is a randomly chosen integer in $[1,m]$. The condition $j = j_{rand}$ ensures that $\boldsymbol{U}_{i,G}$ and $\boldsymbol{X}_{i,G}$ differ by at least one parameter. In this paper the components of the trial vector $\boldsymbol{U}_{i,G}$, are forced to remain within the interval defined in Eq. (9.13), so if the cross-over provides $u^j_{i,G} \leq \boldsymbol{x^L}$ or $u^j_{i,G} \leq \boldsymbol{x^H}$ then $u^j_{i,G}$ is randomly regenerated in the interval of Eq. (9.13).

The new individual $X_{i,G+1}$ is thus evaluated comparing $\boldsymbol{U}_{i,G}$ and $\boldsymbol{X}_{i,G}$ as follows:

$$\boldsymbol{X}_{i,G+1} = \begin{cases} \boldsymbol{U}_{i,G}, & \text{if } J_{U_{i,G}} \leq J_{X_{i,G}}\,, \\ \boldsymbol{X}_{i,G}, & \text{otherwise}\,. \end{cases} \tag{9.19}$$

To clarify, the trial vector is chosen as new individual if its performance index, $J_{U_{i,G}}$, is less or equal to the index value of corresponding target vector, $J_{X_{i,G}}$. The DE algorithm runs until a user-defined convergence criterion is satisfied (a maximum number of iteration is also imposed in case the convergence criterion cannot be satisfied).

9.5 Implementation details

The problem taken into account in this chapter is solved with a differential flatness approach. The flat output is identified with the displacement of the satellites, $\boldsymbol{\rho}$, as $\dot{\boldsymbol{\rho}}$ and $\ddot{\boldsymbol{\rho}}$ may be obtained evaluating the derivatives of the approximating polynomials for $\boldsymbol{\rho}$. In this case, only the three components of $\boldsymbol{\rho}$ will directly enter into the optimization process, thus reducing the number of optimization variables to N_u. The advantages of the DF approach have been presented in Sec. 2.6. The DE-based numerical method has the same fundamental characteristics of the PSO-based optimizer described in Chapter 4.

The trajectory of the chaser can be obtained using different interpolating functions. Once a polynomial approximation for $\boldsymbol{\rho}$ is chosen, its first and second time derivatives may be easily obtained. The most important issue is what kind of polynomial approximation one can choose as this choice can strongly affect the results of the optimization. In this work two different interpolating functions are considered: *Chebyshev polynomials* and *B-spline curves*.

In this section a brief description of the *Chebyshev polynomials* and the *B-spline curves* is provided in Sec. 9.5.1 and Sec. 9.5.2, respectively. Additional details concerning the composition of DE individuals, the definition of the performance index and the convergence criterion are provided in Sec. 9.5.3.

The usage of the Chebyshev polynomials is suggested since they are particularly suitable for minimax problems [188, 189], i.e. when the approximation of a mathematical function $f(x)$, $x \in [a, b]$, is given by solving

$$\min \left(\max_{a \leq x \leq b} |f(x) - p(x)| \right), \tag{9.20}$$

where $p(x)$ is the approximation polynomial. On the other side, the B-spline have been proved to work particularly efficiently for inverse dynamics problems related to attitude problems, as we have shown in the previous chapters.

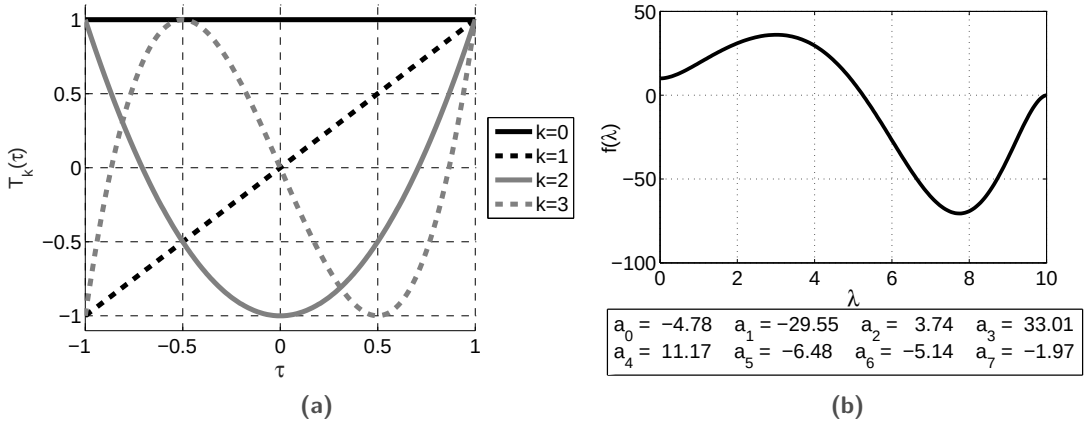


Fig. 9.1: Chebyshev polynomials (a) and example of Chebyshev curve (b) with $N_{\mathcal{P}} = 8$.

9.5.1 Chebyshev approximation

Chebyshev polynomials of first kind represent a series of orthogonal polynomials defined over the interval $-1 \leq \tau \leq 1$. Accordingly, the one-to-one mapping

$$\tau = 2\,\frac{t - t_0}{t_f - t_0} - 1 \tag{9.21}$$

from the original time domain $[t_0, t_f]$ to the Chebyshev domain $[\tau_0, \tau_f] = [-1, 1]$ is employed. The zeroth and the first order Chebyshev polynomial are defined as $T_0(\tau) = 1$ and $T_1(\tau) = \tau$, respectively, where the subscripts 0 and 1 are the degrees of the polynomials. Higher degree polynomials are then obtained with the recursive relationship [187]

$$T_{i+1}(\tau) = 2\,\tau T_i(\tau) - T_{i-1}(\tau)\,. \tag{9.22}$$

In Fig. 9.1(a) the Chebyshev polynomials from degree 0 to degree 3 are illustrated. It can be observed that the range of the Chebyshev polynomials is equal to their domain, i.e. $-1 \leq T_i(\tau) \leq 1 \;\; \forall i$.

First and second derivatives of the Chebyshev polynomials can be easily evaluated with the following recursive relationships:

$$T'_{i+1}(\tau) = \frac{dT_{i+1}(\tau)}{d\tau} = 2\tau T'_i(\tau) + 2T_i(\tau) - T'_{i-1}(\tau) \tag{9.23}$$

$$T''_{i+1}(\tau) = \frac{d^2T_{i+1}(\tau)}{d\tau^2} = iT'_i(\tau) + \frac{T''_{i-1}(\tau)}{i-2} \tag{9.24}$$

where $(\cdot)'$ is the derivative with respect to τ. The Chebyshev approximation $\mathcal{C}$ for the i^{th} component of the relative displacement ρ_i is given as a Chebyshev series truncated at $N_{\mathcal{P}}$ polynomials and defined upon $\tau(t)$ as

$$\rho_i(\tau) = \mathcal{C}(\tau; \boldsymbol{a}) = \sum_{i=0}^{N_{\mathcal{P}}-1} a_i T_i(\tau)\,, \tag{9.25}$$

where $\boldsymbol{a}$ is the coefficients vector for the polynomial combination and it is given by

$$\boldsymbol{a} = [a_0, a_1, ..., a_{N_{\mathcal{P}}-1}]^T\,. \tag{9.26}$$

The first and the second time derivatives of ρ_i are given as

$$\dot{\rho}_i = \dot{\mathcal{C}}(\tau; \boldsymbol{a}) = \Delta_\tau \sum_{i=1}^{N_{\mathcal{P}}-1} a_i T_i'(\tau)\;,\quad \ddot{\rho}_i = \ddot{\mathcal{C}}(\tau; \boldsymbol{a}) = \Delta_\tau^2 \sum_{i=2}^{N_{\mathcal{P}}-1} a_i T_i''(\tau)\,. \tag{9.27}$$

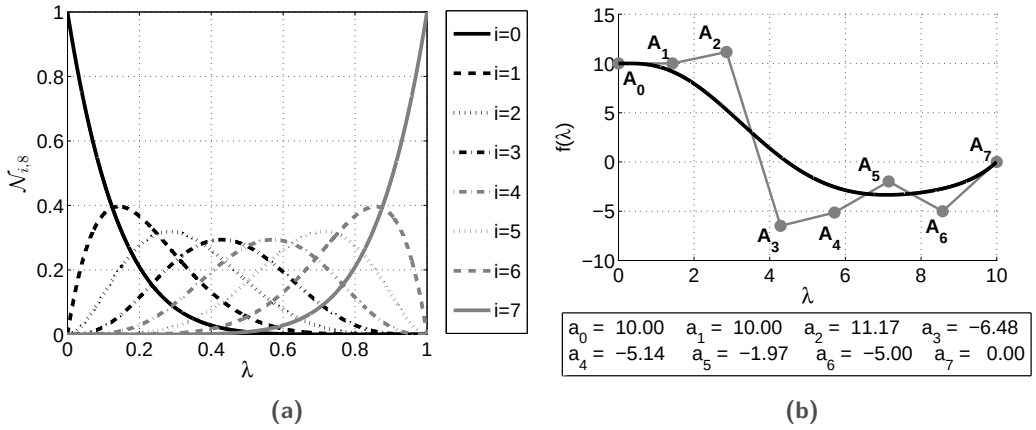
Note that $\dot{(\cdot)}$ is the derivative with respect to t: the parameter $\Delta_\tau = d\tau/dt$ takes into account the change of domain and, given Eq. (9.21), is evaluated as

$$\Delta_\tau = \frac{d\tau}{dt} = \frac{2}{t_f - t_0}\,. \tag{9.28}$$

The Chebyshev coefficients represent the optimization variables. The inverse dynamics approach require to fix some optimization coefficients to satisfy all the boundary conditions. In our case, referring to Eq. (9.5), position and velocity are constrained at t_0 and t_f. Accordingly, for every axis 4 conditions must be imposed by fixing 4 coefficients (the coefficients to be fixed may be freely chosen among all the optimization variables). For the generic maneuver axis, the coefficients a_0, a_1, a_2 and a_3 are obtained as

$$\begin{aligned}
a_0 &= \frac{x_f + x_0}{2} - \sum_{i=2,4,...}^{N_{\mathcal{P}}-1} a_i\,,\\
a_1 &= \frac{x_f - x_0}{2} - \sum_{i=3,5,...}^{N_{\mathcal{P}}-1} a_i\,,\\
a_2 &= \frac{\dot{x}_f - \dot{x}_0}{8} - \sum_{i=4,6,...}^{N_{\mathcal{P}}-1} \left(\frac{i}{2}\right)^2 a_i\,,\\
a_3 &= \frac{\dot{x}_f + \dot{x}_0 - x_f + x_0}{16} - \sum_{i=5,7,...}^{N_{\mathcal{P}}-1} \binom{i-2}{2} a_i\,.
\end{aligned} \tag{9.29}$$

In Fig. 9.1(b) a curve obtained with a Chebyshev series is shown using $N_{\mathcal{P}} = 8$ and $t_f = 10$ time units. The values of the first 4 coefficients have been chosen in order to set $\mathcal{C}(-1; \boldsymbol{a}) = 10$, $\mathcal{C}(1; \boldsymbol{a}) = 0$, $\dot{\mathcal{C}}(-1; \boldsymbol{a}) = 0$ and $\dot{\mathcal{C}}(1; \boldsymbol{a}) = 3.5\,\Delta_\tau$.



(a) (b)

Fig. 9.2: Basis functions $\mathcal{N}_{i,\mathcal{D}}$(a) and example of clamped B-spline curve (b) with $N_{\mathcal{P}} = \mathcal{D} = 8$.

9.5.2 B-splines approximation

The description of the B-spline approximation is carried out in Sec. 4.2. In this chapter, a comparison between the basic and the improved models is carried out.

In Fig. 9.2(a) the basis functions $\mathcal{N}_{i,\mathcal{D}}$ are shown having set $N_{\mathcal{P}} = \mathcal{D} + 1 = 8$. A basic B-spline curve is reported in Fig. 9.2(b) for $t_f = 10$ time units; as for Fig. 9.1(b), the coefficients have been chosen in order to set $\mathcal{B}(0;\boldsymbol{a}) = 10$, $\mathcal{B}(1;\boldsymbol{a}) = 0$, $\dot{\mathcal{B}}(0;\boldsymbol{a}) = 0$ and $\dot{\mathcal{B}}(1;\boldsymbol{a}) = 3.5\,\Delta_\lambda$. Through explanations may be found in [76, 77, 78].

9.5.3 Optimizer setup

A generic DE individual is defined as

$$\boldsymbol{X} = [\boldsymbol{a}_x, \boldsymbol{a}_y, \boldsymbol{a}_z,\ t_f]\,, \tag{9.30}$$

where the vector $\boldsymbol{a}_j = [a_{j,0}, ..., a_{j,\mathcal{N}_p-1}]$ is defined as in Eq. (9.26) and contains $N_{\mathcal{P}}$ interpolation parameters for the axis j that can be used as Chebyshev coefficients or as y coordinate of the B-spline control points. When using the improved B-spline approximation, the vectors $\boldsymbol{b}_j,\ j = x, y, z$ must be added to the DE individuals. Note that the number of optimization parameters changes considering planar of three-dimensional maneuvers. Let us also repeat that not all the $N_{\mathcal{P}}$ coefficients in $\boldsymbol{a}_j$ are optimization parameters since certain coefficients are enforced to satisfy the boundary conditions. For the numerical simulations reported in the sequel, the trajectories given by the Chebyshev polynomials and the B-splines curves are discretized over N_t points, which means that time interval $[t_0, t_f]$ is split into N_t instants.

A performance index J is associated to each individual to evaluate the quality of the obtained maneuver. As for the previous chapters, also in this case the performance index is given by an exterior penalty function method (see Sec. 1.3.2). Accordingly, J is calculated as

$$J = \tilde{\alpha} t_f + \tilde{\beta} \sum_{j=1}^{3} \sum_{k=0}^{N_t} \eta_{j,k}(\boldsymbol{a}_j, t_k) + \tilde{n} N_{viol} \,, \tag{9.31}$$

where $\tilde{\alpha}$, $\tilde{\beta}$ and $\tilde{n}$ are user-defined constant and t_f is the time required to complete the maneuver. The second term is the control penalty function, and it penalises the solutions for which the control exceeds the allowed maximum value u_{max}. Therefore, $\eta_{j,k}(\boldsymbol{a}_j, t_k)$ is given by

$$\eta_{j,k}(\boldsymbol{a}_j, t_k) = \begin{cases} 0 & \text{if } \dfrac{|u_{j,k}(\boldsymbol{a}_j, t_k)|}{u_{max}} \leq 1 \,, \\ \dfrac{|u_{j,k}(\boldsymbol{a}_j, t_k)|}{u_{max}} - 1 & \text{otherwise} \,. \end{cases} \tag{9.32}$$

Note that the terms $u_{j,k}$ are the discretized values obtained from Eq. (9.2). Finally, N_{viol} represents the number of violated constraints, and it awards individuals which violates less constraints than other.

Denoting with $t_{f,i}^{best}$ the maneuver time associated to the best solution at generation i, the iterations stop when the standard deviation of the lasts ten best maneuver times is less than the user-defined tolerance ϵ, that is:

$$\sqrt{\frac{\sum_{i=1}^{10} \left(t_{f,i}^{best} - \bar{t}_f^{best} \right)^2}{10}} \leq \epsilon \,. \tag{9.33}$$

9.6 Numerical results

In this section the performances of proposed techniques are evaluated by means of several example cases. In all the numerical experiments, maneuvers between a controlled deputy and a non-cooperative chief spacecraft are taken into account. The orbital parameters of the chief satellite are reported in Table 9.2. Non-dimensional units are used; consequently, distances are divided by K_x, that is the initial relative position between both satellites, and the control is divided by $K_u = u_{max}$. The maneuver time is normalized by $K_t = \sqrt{K_x / u_{max}}$, and finally velocities are divided by $K_v = K_x / K_t$. The maximum value of the thrust has been set to $u_{max} = 5 \cdot 10^{-4}$ m/s^2.

Tab. 9.2: Orbital parameters of the chief satellite.

Parameter	Value
a	7000 km
e	0
i	45 deg
Ω	0
ω	0

The DE parameters used in the numerical simulations are reported in Tables 9.3 and 9.4, where the former shows the constant parameters and the latter provides the parameters that linearly change during the optimization process. The scale factors for the local and the global mutations have the same values, i.e. $F_l = F_g = F$. The choice of these parameters has been carried out after having run several simulations and having compared the results. The parameters of the performance index in Eq. (9.31) are $\tilde{\alpha} = \tilde{\beta} = 1$ while $\tilde{n} = 100$.

With regard to the parameters concerning the definition of the interpolation, the number of control points and the degree of the basis polynomials have been chosen after some preliminary analysis. For the Chebyshev interpolation, the motion is approximated using $\mathcal{N}_p = 8$ polynomials going from degree zero to degree seven. In order to respect the initial and final conditions, the coefficients $a_{j,0}$, $a_{j,1}$, $a_{j,2}$, $a_{j,3}$ are given by Eq. (9.29). For the B-spline, $\mathcal{N}_p = 8$ control points are used, with Basis functions of seven th degree. In this case, the a-priori computed parameters are $a_{j,0}$, $a_{j,1}$, $a_{j,\mathcal{N}_p-2}$, $a_{j,\mathcal{N}_p-1}$, according to Eq. (4.20). In both cases, the free optimization parameters of the DE individual in Eq. (9.30) are $3(\mathcal{N}_p - 4) + 1$, i.e. the free interpolation coefficients plus the final time (for the improved B-spline approximation the optimization parameters are $6(\mathcal{N}_p - 2) - 2$, since the values of $b_{i,j},\ i = x, y, z,\ j = 2, ..., \mathcal{N}_p$ must be considered). The trajectories are discretized over $N_t = 200$ points.

The proposed guidance algorithm is verified through several simulation campaigns. First, an analysis for a thorough comparison between the Chebyshev and the B-spline

Tab. 9.3: Differential evolution constant parameters.

Parameter	Value
$\mathcal{S}$	50
CR	0.75
F	0.8
R_L	3
N_{max}	10,000
x^L	-5
x^H	-5

Tab. 9.4: Differential evolution variable parameters.

Parameter	Initial Value (1^{st} step)	Final Value (2500^{th} step)
λ	0.9	0.1
w	0	1

approximation methods is performed in Sec. 9.6.1. Autonomous docking maneuvers will be taken into account to accomplish this task. Second, a comparison of the proposed DE-based optimizer with an analogous PSO-based optimizer is performed in Sec. 9.6.2. Lastly, Sec. 9.6.3 deals with some reconfiguration maneuvers considering reference trajectories given in the SS dynamical model.

9.6.1 Performance analysis

This first test case is intended to compare the Chebyshev and the B-spline approximation methods in order to understand what kind of performances they achieve and consequently choose for the best approach. For this case, the planar motion only is considered, i.e. the motion along $Z_{\mathscr{L}}$ is not controlled.

To perform the proposed comparison, a Monte Carlo simulation based on $10,000$ test cases has been run. Docking maneuvers are taken into account. The deputy must reach the chief, which is at the center of the LVLH reference frame, starting from initial conditions that have been randomly generated and uniformly distributed between a minimum and maximum distances of 0.1 km and 2 km, respectively. All the initial velocities along the x axis have been initialized in the range $[10^{-2}, 10^{-5}]$ km/s. With regard to the velocities along the y axis, all the imposed initial conditions satisfy the no along-track shift condition [132],

$$\dot{y}(t_0) = -\beta x(t_0) \tag{9.34}$$

which is valid in the SS model where $\beta = 2\bar{m}$. Imposing $\beta = 2\omega$, initial conditions representing quasi-equilibrium states can be set, since the optimization process is carried out considering the J_2-perturbed model. In this way, the ability of the external control to compensate for little initial drifts due to imperfect initial conditions is investigated. Moreover, the 10,000 simulation have been equally distributed inside four circular crowns given by the following initial distances: $r \in (0.1, 0.5]$ km, $r \in (0.5, 1.0]$ km, $r \in (1.0, 1.5]$ km and $r \in (1.5, 2.0)$ km. The introduction of these circular crowns will help evaluating the performances of the algorithm varying the initial distance of the deputy. In Fig. 9.3 the comparison between the performances of the two approximation approaches is reported. The reported curves have been obtained fitting with a second order polynomial the Monte Carlo results. In Fig.

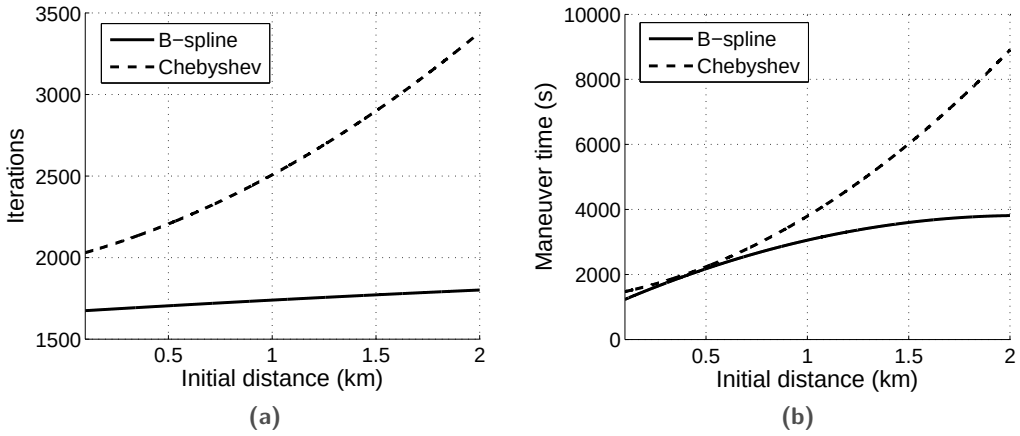


Fig. 9.3: Comparison of Chebyshev and B-spline approximations performances.

9.3.(a) the required number of iterations is shown for a value of ϵ in Eq. (9.33) equal to 10^{-10}. It can be seen that using the Chebyshev polynomials a greater number of iterations is required with respect to the B-spline approximation, and the difference between the two methods increases as the initial distance of the deputy increases. The number of iterations required by the B-spline approximation increases very slowly whereas the Chebyshev approximation shows a parabolic behavior with a positive second derivative. Moreover, looking at Fig. 9.3.(b) where the maneuver times associated with the optimized trajectories is reported, it can be noted that the B-spline approximation always guarantees lower maneuver times with respect to the Chebyshev approximation. The curves reported in Fig. 9.3 have been obtained by discarding the test cases where the maximum number of allowed iterations has been reached (196 cases for the Chebyshev polynomials and only 2 cases and the B-spline). In addition Fig 9.4 shows the distributions in the number of iterations required to reach the sub-optimal solution both with Chebyshev approximation (Fig 9.4.(a)) and B-splines approximation (Fig. 9.4.(b)). First of all it can be noted that the two distributions are not normal distributions, but in both cases Fig. 9.4 provides a log-normal distribution,which means that the occurrances are not distribuited symmetrically in relation to their means. The parameters which describe the log-normal distribution are log-scale,μ_L and log-scale, σ_L, by which it is possible to obtain mean and variance with the folowing relations:

$$\mu = e^{\mu_L + \frac{\sigma_L^2}{2}} \tag{9.35}$$

$$\sigma^2 = e^{2\mu_L + \sigma_L^2}(e^{\sigma_L^2} - 1) \tag{9.36}$$

By replacing the parameters of both Chebyshev and B-spline distributions in (9.35) and (9.36), it is possible to obtain their arithmetic means and arithmetic standard deviation for B-spline which are $\mu_{BS} = 1.73 \cdot 10^3$ and $\sigma_{BS} = 2.59 \cdot 10^2$ respectively, while for Chebyshev approximation $\mu_{Cheb} = 2.16 \cdot 10^3$ and $\sigma_{Cheb} = 4.01 \cdot 10^2$

respectively. Summarizing, it can be note that the number of iterations required to reach sub-optimal solution with B-splines, are, on average, lower compared to Chebyshev, and, in addition, even their dispersion in relation to the average is the lowest.

The reason why the B-splines approximation outperforms the Chebyshev approximation is the better ability of the former to catch the ideal bang-bang solution given in Sec. 9.2 with respect to the latter. Fig. 9.5 reports the control histories associated to a maneuver with initial conditions equal to $[x(0),\ y(0),\ \dot{x}(0),\ \dot{y}(0)] = [-1,\ 1,\ 0,\ -2\omega]$ for both the approximation methods. The B-spline maneuver requires 3988 seconds while the Chebyshev maneuvers requires 4113 seconds. It can be noted that the B-spline approximation (Fig. 9.5.(b)) is able to catch a good approximation of the bang-bang behavior both for the x and the y axis. In this case, the sequences of bangs $[+1, -1, +1]$ for the x axis and $[+1, -1+1, -1]$ for the y axis seem to be optimal. The Chebyshev approximation (Fig. 9.5.(a)) cannot provide a good approximation for u_x, while u_y is quite the same as the one obtained with the B-spline. The trajectories obtained with the two control policies are shown in Fig. 9.6. It is noteworthy that the B-spline trajectory brings the deputy satellite toward greater values in positive direction of the x axis. In this way, the difference between the absolute velocity of deputy and chief satellites induced by the Keplerian gravity effect is maximized, which is consistent with the achieved minimum-time maneuver.

To study in more detail the differences between the B-spline and the Chebyshev approximation schemes, let us consider the results reported in Table 9.5 and Table 9.6. Here, the performances of the proposed algorithm are evaluated considering different values for the convergence threshold ϵ. The parameter N_ϵ is the number of

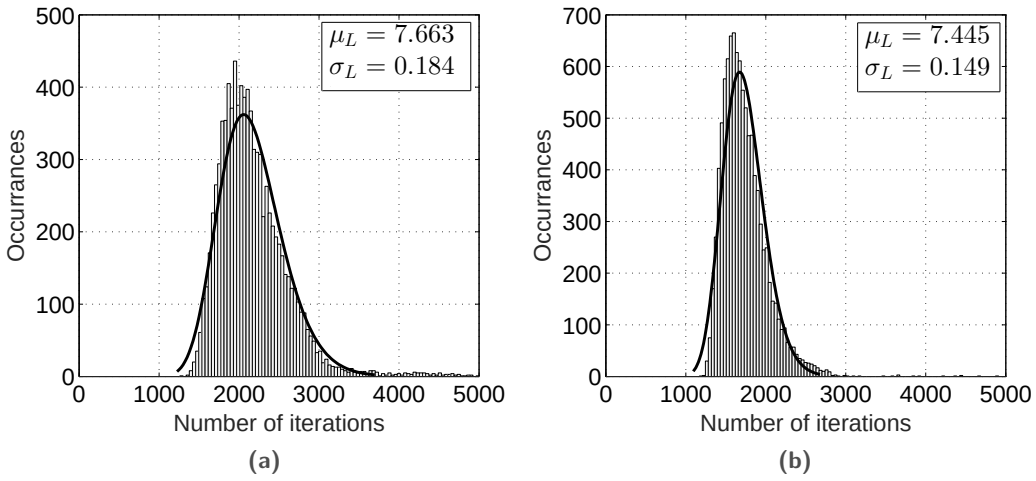


Fig. 9.4: Chebyshev (a) and B-splines (b) iterations number with log-normal interpolation.

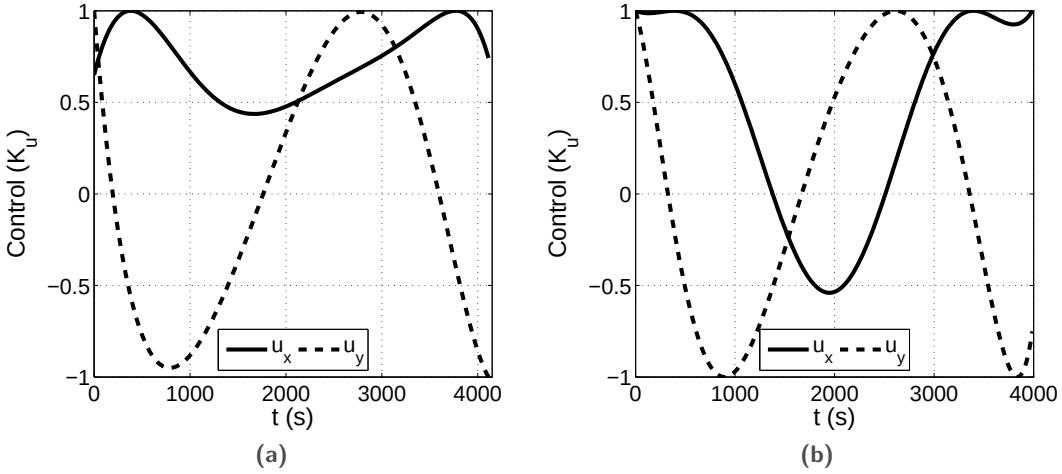


Fig. 9.5: Control policy obtained with Chebyshev (a) and B-spline (b).

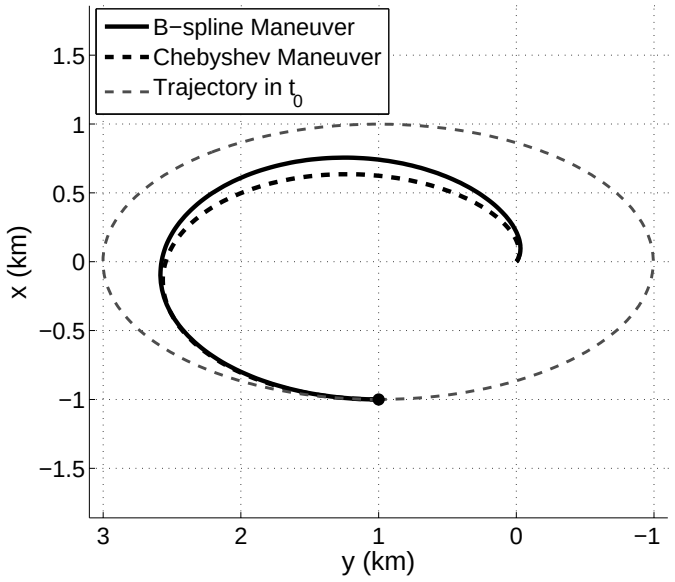


Fig. 9.6: Trajectories obtained with Chebyshev and B-spline.

iterations required to satisfy the tolerance ϵ. The relative difference Δt_f is evaluated as

$$\Delta t_{f,\epsilon} = \frac{t_{f,\epsilon} - t_f^*}{t_f^*} \tag{9.37}$$

where $t_{f,\epsilon}$ is the maneuver time associated with the tolerance ϵ and t_f^* is the maneuver time associated with $\epsilon = 10^{-10}$. Comparing the results in the two tables, notes are:

1. As already pointed out in Fig. 9.3, the number of iterations using the B-spline approximation is always fewer than the one required by the Chebyshev approximation.

Tab. 9.5: Results of Monte Carlo simulation with the Chebyshev approximation.

	$\epsilon = 10^{-6}$	$\epsilon = 10^{-7}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-9}$	$\epsilon = 10^{-10}$
			$r \in (0.1, 0.5]$ km		
Mean(N_ϵ)	$1.39{\cdot}10^{3}$	$1.54{\cdot}10^{3}$	$1.71{\cdot}10^{3}$	$1.89{\cdot}10^{3}$	$2.14{\cdot}10^{3}$
Std(N_ϵ)	$2.03{\cdot}10^{2}$	$2.44{\cdot}10^{2}$	$3.69{\cdot}10^{2}$	$4.62{\cdot}10^{2}$	$8.66{\cdot}10^{2}$
Mean($\Delta t_{f,\epsilon}$)	$3.95{\cdot}10^{-3}$	$3.83{\cdot}10^{-3}$	$2.88{\cdot}10^{-3}$	$1.45{\cdot}10^{-3}$	0.00
Std($\Delta t_{f,\epsilon}$)	$7.53{\cdot}10^{-2}$	$7.53{\cdot}10^{-2}$	$6.97{\cdot}10^{-2}$	$3.08{\cdot}10^{-2}$	0.00
Max$\Delta t_{f,\epsilon}$	3.11	3.11	3.11	1.38	0.00
			$r \in (0.5, 1.0]$ km		
Mean(N_ϵ)	$1.47{\cdot}10^{3}$	$1.65{\cdot}10^{3}$	$1.83{\cdot}10^{3}$	$2.05{\cdot}10^{3}$	$2.38{\cdot}10^{3}$
Std(N_ϵ)	$2.62{\cdot}10^{2}$	$3.17{\cdot}10^{2}$	$4.29{\cdot}10^{2}$	$6.86{\cdot}10^{2}$	$1.23{\cdot}10^{3}$
Mean($\Delta t_{f,\epsilon}$)	$2.32{\cdot}10^{-2}$	$2.28{\cdot}10^{-2}$	$2.11{\cdot}10^{-2}$	$1.50{\cdot}10^{-2}$	0.00
Std($\Delta t_{f,\epsilon}$)	$2.10{\cdot}10^{-1}$	$2.10{\cdot}10^{-1}$	$2.05{\cdot}10^{-1}$	$1.71{\cdot}10^{-1}$	0.00
Max($\Delta t_{f,\epsilon}$)	3.66	3.66	3.66	3.66	0.00
			$r \in (1.0, 1.5]$ km		
Mean(N_ϵ)	$1.57{\cdot}10^{3}$	$1.76{\cdot}10^{3}$	$1.99{\cdot}10^{3}$	$2.26{\cdot}10^{3}$	$2.67{\cdot}10^{3}$
Std(N_ϵ)	$3.69{\cdot}10^{2}$	$5.43{\cdot}10^{2}$	$7.62{\cdot}10^{2}$	$1.09{\cdot}10^{3}$	$1.62{\cdot}10^{3}$
Mean($\Delta t_{f,\epsilon}$)	$8.74{\cdot}10^{-2}$	$7.46{\cdot}10^{-2}$	$5.28{\cdot}10^{-2}$	$2.94{\cdot}10^{-2}$	0.00
Std($\Delta t_{f,\epsilon}$)	$5.49{\cdot}10^{-1}$	$5.01{\cdot}10^{-1}$	$3.93{\cdot}10^{-1}$	$2.55{\cdot}10^{-1}$	0.00
Max($\Delta t_{f,\epsilon}$)	9.85	9.85	7.19	4.85	0.00
			$r \in (1.5, 2.0)$ km		
Mean(N_ϵ)	$1.68{\cdot}10^{3}$	$1.93{\cdot}10^{3}$	$2.21{\cdot}10^{3}$	$2.56{\cdot}10^{3}$	$3.09{\cdot}10^{3}$
Std(N_ϵ)	$5.49{\cdot}10^{2}$	$8.37{\cdot}10^{2}$	$1.17{\cdot}10^{3}$	$1.56{\cdot}10^{3}$	$2.09{\cdot}10^{3}$
Mean($\Delta t_{f,\epsilon}$)	$1.88{\cdot}10^{-1}$	$1.64{\cdot}10^{-1}$	$1.17{\cdot}10^{-1}$	$7.14{\cdot}10^{-2}$	0.00
Std($\Delta t_{f,\epsilon}$)	$7.89{\cdot}10^{-1}$	$7.29{\cdot}10^{-1}$	$5.81{\cdot}10^{-1}$	$4.48{\cdot}10^{-1}$	0.00
Max($\Delta t_{f,\epsilon}$)	9.05	9.01	6.39	6.39	0.00

2. The mean and the maximum values of $\Delta t_{f,\epsilon}$ for the B-spline are much lower than for Chebyshev. This means that the B-spline approximation can converge suddenly to the optimal maneuver, while the Chebyshev approximation can get stalled over local minima and reach the best solution only increasing the number of iterations. Moreover, the result shown for the max value of $\Delta t_{f,\epsilon}$ when $r \in (0.5, 1.0]$ km in Table 9.5 indicates that the Chebyshev approximation risks to converge toward high maneuver time, which is confirmed by Fig. 9.3.

3. As a result of points 1. and 2., low values of the threshold ϵ may be used with the B-spline approximation when a good, feasible maneuver is searched reducing the computational time at most.

[

To conclude the comparison between the two approximation methods, consider the results reported in Tables 9.7, 9.8, 9.9. Here the mean number and the standard deviation of iterations and the maneuver time errors are reported for the first *feasible*

Tab. 9.6: Results of Monte Carlo simulation with the B-spline approximation.

	$\epsilon = 10^{-6}$	$\epsilon = 10^{-7}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-9}$	$\epsilon = 10^{-10}$
			$r \in (0.1, 0.5]$ km		
Mean(N_ϵ)	$1.08{\cdot}10^{3}$	$1.23{\cdot}10^{3}$	$1.39{\cdot}10^{3}$	$1.54{\cdot}10^{3}$	$1.69{\cdot}10^{3}$
Std(N_ϵ)	$1.25{\cdot}10^{2}$	$1.50{\cdot}10^{2}$	$1.83{\cdot}10^{2}$	$2.22{\cdot}10^{2}$	$2.66{\cdot}10^{2}$
Mean($\Delta t_{f,\epsilon}$)	$1.96{\cdot}10^{-5}$	$9.31{\cdot}10^{-6}$	$5.37{\cdot}10^{-6}$	$2.89{\cdot}10^{-6}$	0.00
Std ($\Delta t_{f,\epsilon}$)	$9.03{\cdot}10^{-5}$	$7.61{\cdot}10^{-5}$	$6.92{\cdot}10^{-5}$	$6.54{\cdot}10^{-5}$	0.00
Max($\Delta t_{f,\epsilon}$)	$3.01{\cdot}10^{-3}$	$3.01{\cdot}10^{-3}$	$3.01{\cdot}10^{-3}$	$3.01{\cdot}10^{-3}$	0.00
			$r \in (0.5, 1.0]$ km		
Mean(N_ϵ)	$1.12{\cdot}10^{3}$	$1.28{\cdot}10^{3}$	$1.41{\cdot}10^{3}$	$1.57{\cdot}10^{3}$	$1.74{\cdot}10^{3}$
Std(N_ϵ)	$1.32{\cdot}10^{2}$	$1.63{\cdot}10^{2}$	$1.92{\cdot}10^{2}$	$2.41{\cdot}10^{2}$	$3.54{\cdot}10^{2}$
Mean($\Delta t_{f,\epsilon}$)	$3.38{\cdot}10^{-5}$	$2.00{\cdot}10^{-5}$	$1.37{\cdot}10^{-5}$	$8.31{\cdot}10^{-6}$	0.00
Std ($\Delta t_{f,\epsilon}$)	$2.49{\cdot}10^{-4}$	$2.36{\cdot}10^{-4}$	$2.25{\cdot}10^{-4}$	$2.11{\cdot}10^{-4}$	0.00
Max($\Delta t_{f,\epsilon}$)	$1.01{\cdot}10^{-2}$	$9.91{\cdot}10^{-3}$	$9.91{\cdot}10^{-3}$	$9.49{\cdot}10^{-3}$	0.00
			$r \in (1.0, 1.5]$ km		
Mean(N_ϵ)	$1.13{\cdot}10^{3}$	$1.28{\cdot}10^{3}$	$1.43{\cdot}10^{3}$	$1.58{\cdot}10^{3}$	$1.74{\cdot}10^{3}$
Std(N_ϵ)	$1.36{\cdot}10^{2}$	$1.63{\cdot}10^{2}$	$2.02{\cdot}10^{2}$	$2.38{\cdot}10^{2}$	$3.24{\cdot}10^{2}$
Mean($\Delta t_{f,\epsilon}$)	$4.96{\cdot}10^{-5}$	$3.15{\cdot}10^{-5}$	$1.87{\cdot}10^{-5}$	$1.30{\cdot}10^{-5}$	0.00
Std ($\Delta t_{f,\epsilon}$)	$4.32{\cdot}10^{-4}$	$4.02{\cdot}10^{-4}$	$3.77{\cdot}10^{-4}$	$3.63{\cdot}10^{-4}$	0.00
Max($\Delta t_{f,\epsilon}$)	$1.73{\cdot}10^{-2}$	$1.72{\cdot}10^{-2}$	$1.72{\cdot}10^{-2}$	$1.67{\cdot}10^{-2}$	0.00
			$r \in (1.5, 2.0)$ km		
Mean(N_ϵ)	$1.17{\cdot}10^{3}$	$1.32{\cdot}10^{3}$	$1.47{\cdot}10^{3}$	$1.62{\cdot}10^{3}$	$1.79{\cdot}10^{3}$
Std(N_ϵ)	$1.67{\cdot}10^{2}$	$1.94{\cdot}10^{2}$	$2.22{\cdot}10^{2}$	$3.26{\cdot}10^{2}$	$4.70{\cdot}10^{2}$
Mean($\Delta t_{f,\epsilon}$)	$9.41{\cdot}10^{-4}$	$9.06{\cdot}10^{-4}$	$8.79{\cdot}10^{-4}$	$5.31{\cdot}10^{-4}$	0.00
Std ($\Delta t_{f,\epsilon}$)	$1.87{\cdot}10^{-2}$	$1.85{\cdot}10^{-2}$	$1.84{\cdot}10^{-2}$	$1.63{\cdot}10^{-2}$	0.00
Max($\Delta t_{f,\epsilon}$)	$7.49{\cdot}10^{-1}$	$7.49{\cdot}10^{-1}$	$7.49{\cdot}10^{-1}$	$7.49{\cdot}10^{-1}$	0.00

Tab. 9.7: Feasibility analysis from the Monte Carlo results, mean values.

Circular crown (km)	*Chebyshev approximation* Mean(N_F) (-)	Mean($\Delta t_{f,F}$) (-)	Max($\Delta t_{f,F}$) (-)
$r \in (0.1, 0.5]$	95.50	23.48	97.73
$r \in (0.5, 1.0]$	218.20	12.41	96.72
$r \in (1.0, 1.5]$	395.05	8.00	73.97
$r \in (1.5, 2.0)$	676.18	6.15	57.21

maneuver, i.e. the first maneuver that satisfies all the imposed constraints. The feasible maneuver is reached after N_F iterations, and the relative maneuver time difference $\Delta t_{f,F}$ is evaluated as in Eq. (9.37). Note that, since these results are related to the very first part of the optimization algorithm, the standard deviation values reported in Table 9.9 are quite high. Nonetheless, the B-spline values are lower than the Chebyshev ones, showing that also in this case the former is more stable and reliable than the latter. Note also that, for both approximation schemes,

!t]

Tab. 9.8: Feasibility analysis from the Monte Carlo results, mean values.

Circular crown (km)	*B-spline approximation* Mean(N_F) (-)	Mean($\Delta t_{f,F}$) (-)	Max($\Delta t_{f,F}$) (-)
$r \in (0.1, 0.5]$	11.32	8.83	39.7
$r \in (0.5, 1.0]$	54.97	5.51	24.35
$r \in (1.0, 1.5]$	105.40	3.97	20.68
$r \in (1.5, 2.0)$	165.13	3.14	20.48

Tab. 9.9: Feasibility analysis from the Monte Carlo results, standard deviations.

Circular crown (km)	*Chebyshev approximation* Std(N_F) (-)	Std($\Delta t_{f,F}$) (-)	*B-spline approximation* Std (N_F) (-)	Std ($\Delta t_{f,F}$) (-)
$r \in (0.1, 0.5]$	38.33	12.32	18.91	5.09
$r \in (0.5, 1.0]$	$1.53{\cdot}10^2$	9.81	50.38	4.22
$r \in (1.0, 1.5]$	$5.49{\cdot}10^2$	8.02	81.52	3.93
$r \in (1.5, 2.0)$	$1.05{\cdot}10^3$	7.10	$1.31{\cdot}10^2$	2.82

the feasible maneuver gets closer to the optimal one when the initial distance of the deputy increases.

As described in Sec. 9.5.2, full advantage of the B-spline approximation is obtained when also the time is approximated as a B-spline. In this case, the *graph* of the function is approximated, thus having the opportunity to model the curve both along the vertical and horizontal directions [2, 3]. Let us refer to this approach as *improved* B-spline approximation, in contrast with the previous *basic* B-spline approximation. When using the improved B-spline approximation, the optimal control obtained

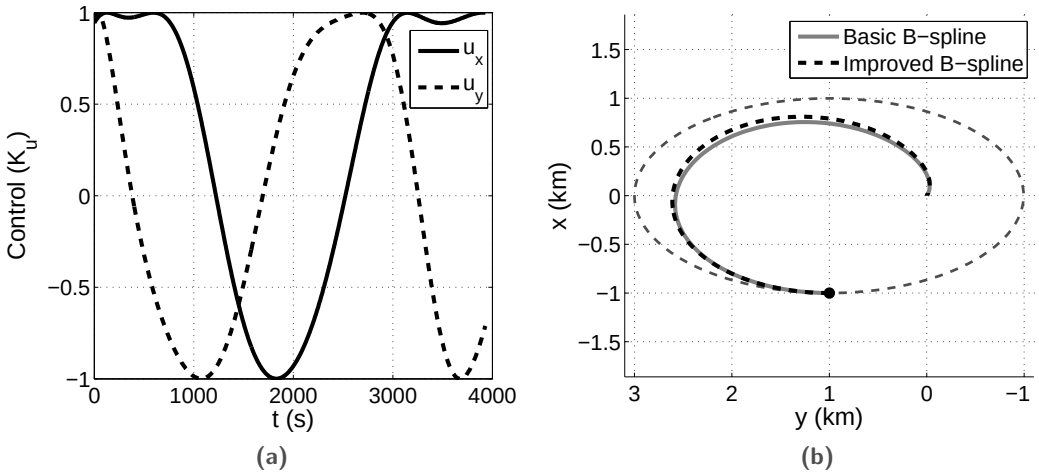


Fig. 9.7: Control policy (a) and trajectory (b) with the improved B-spline approximation.

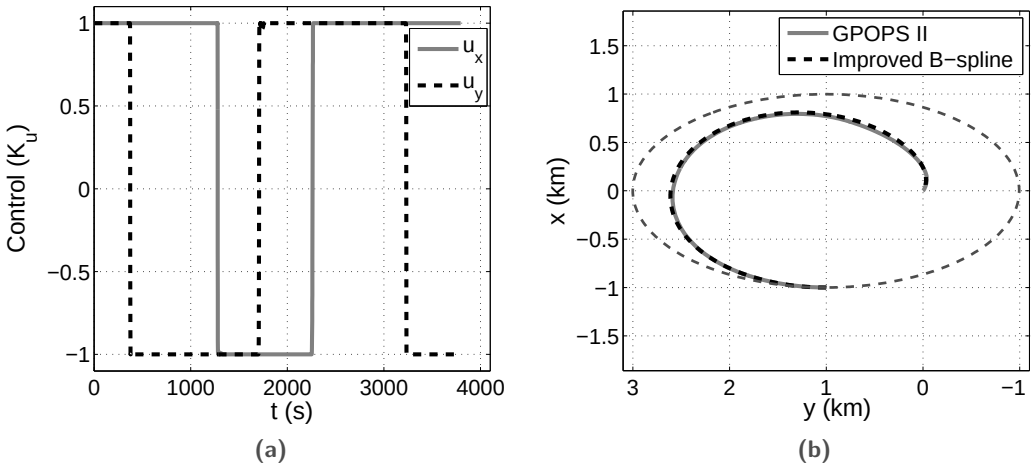


Fig. 9.8: Control policy (a) and trajectory (b) with POCS.

for the previous example of Fig. 9.5, 9.6 is reported in Fig. 9.7. The maneuver time associated with the improved solution is 3930 seconds, which is lower than the 3988 seconds of the basic B-spline solution. Looking at Fig. 9.7.(a), it can be seen that the B-spline is now able to catch a very good approximation of the bang bang solution, even though the trajectory shown in Fig. 9.7.(b) is quite the same than the one obtained with the basic approximation scheme. The efficiency of the improved B-spline approximation may be appreciated looking at the results given by the software GPOPS II [57] (used under research license). As can be seen, the improved B-spline control is a precise approximation of the bang-bang control. Moreover, the GPOPS II and the improved B-spline trajectory overlap quite perfectly. However, the strong disadvantage related to the improved B-spline approximation is that the computational effort and the number of iterations considerably grow up. For example, this test case required 7226 steps, whereas the same case with the basic B-spline approximation required only 2689 iterations. Moreover, the evaluation of the curve is more time-consuming with respect to the basic B-spline approximation. Accordingly, the basic B-spline approximation is generally suggested when searching for a good compromise between optimality and computational cost.

As a result of the reported tables and figures, the B-spline approximation outperforms the Chebyshev approximation and will be used for the following result section. The basic B-spline approximation will be employed to reduce the computational cost of the planning algorithm.

9.6.2 Comparison with the particle swarm optimization

To assess the performances of the proposed DE-based optimization algorithm, a comparison with a PSO-based optimizer is reported. The PSO-based planning

Tab. 9.10: Comparison between the DE and the PSO performances.

Test Case	$x(t_0)$ (km)	$y(t_0)$ (km)	$\dot{x}(t_0)$ (km/s)	$\dot{y}(t_0)$ (km/s)	DE N (-)	DE t_f (10^3 s)	PSO N (-)	PSO t_f (10^3 s)
1	-1.00	$+1.00$	0.00	$+2.16\cdot10^{-2}$	2689	4.02	3139	3.94
2	-0.16	-0.52	$+9.03\cdot10^{-5}$	$+3.45\cdot10^{-4}$	1725	1.57	1531	1.59
3	-0.72	$+0.35$	$+4.86\cdot10^{-4}$	$+1.55\cdot10^{-3}$	1780	2.79	1142	2.88
4	$+0.36$	$+0.13$	$-3.03\cdot10^{-5}$	$-7.85\cdot10^{-4}$	2687	1.89	1651	1.93
5	-0.25	$+1.27$	$+9.37\cdot10^{-5}$	$+5.37\cdot10^{-4}$	1755	3.21	5369	3.39
6	-1.09	$+0.54$	$+6.97\cdot10^{-3}$	$+2.34\cdot10^{-3}$	1951	6.30	1514	11.21
7	-1.61	$+0.34$	$-4.88\cdot10^{-4}$	$+3.47\cdot10^{-3}$	2700	4.67	4297	4.89
8	$+1.15$	-0.01	$-9.30\cdot10^{-3}$	$-2.47\cdot10^{-3}$	10000	5.25	3465	9.92
9	-0.07	-1.58	$-4.81\cdot10^{-4}$	$+1.45\cdot10^{-4}$	2264	2.57	3708	2.56
10	$+1.20$	-1.12	$-3.13\cdot10^{-4}$	$-2.59\cdot10^{-3}$	1746	3.93	2140	3.95

algorithm is exactly the same as the one described in the previous paragraphs, but PSO is used instead of DE for the trajectory optimization. The unified approach with local and global best is employed and 50 particles are involved. The goal of this section is to compare the DE results with those obtained with PSO in order to check that they are in agreement with the literature (in [175, 176, 177] it is stated that DE generally outperforms PSO). However, it is well-known that recognizing the best algorithm between two heuristic approaches is not an easy task (see the discussion about the No Free Launch theorem in [67]). Accordingly, our purpose will mainly be to check the consistency of the DE results.

The results of the comparison are reported in Table 9.10 for 10 test cases taken from the previous Monte Carlo analysis. As can be seen, DE generally outperforms PSO, but the authors believe that such results strongly depend on the parameters governing the behavior of the two algorithms. The maneuver time given by PSO is lower than the DE one only for test cases 1 and 9, while the number of required iterations does not allow to say which algorithm behaves the best. For instance, it can be seen that for the test case 8 DE provides a better solution than PSO, but the required iterations are 10,000, which is the maximum number of allowed iterations. As previously mentioned, the results in Table 9.10 can be used to get the confidence that the DE has good convergence properties in terms of precision and computational cost.

9.6.3 Reconfiguration

Closed-form solutions of the linear relative motion equations with $\boldsymbol{u} = \mathbf{0}$ and no perturbing forces (i.e. using the HCW equations) are reported in [132]: to obtain closed (or periodic) relative trajectories, the initial condition in Eq. (9.34) must be imposed. Similar trajectories may be obtained from the SS model. As shown in literature [132], three characteristic types of closed relative trajectories can

be derived: Along-Track Formation (ATF), General Circular Formation (GCF) and Projected Circular Formation (PCF). In these cases, the general satellite trajectory can be written in magnitude-phase form as

$$\begin{aligned} x_d(t) &= \frac{1}{2} R \sin\left(\bar{n}t + \alpha_0\right) , \\ y_d(t) &= \rho_{y,0} + R \cos\left(\bar{n}t + \alpha_0\right) , \\ z_d(t) &= \frac{\kappa_{zx}}{2} R \sin\left(\bar{n}t + \alpha_0\right) , \end{aligned} \tag{9.38}$$

where R dictates the dimension the trajectory, α_0 is the initial phase angle and $\rho_{y,0}$ is the center of the close trajectory along the y axis. For the PCF, $\kappa_{zx} = 2$, for the GCF $\kappa_{zx} = \sqrt{3}$ whereas for the ATF $R = 0$ and $\rho_{y,0} \neq 0$. Eq. (9.38) applies for the SS model (with ω instead of $\bar{n}$ they are valid for the HCW model).

In this section, the proposed planner has been used to minimize the time of a reconfiguration maneuver between two spacecrafts in formation. The following maneuvers will be taken into account:

1. ATF to ATF, starting from -0.4 km and arriving at -1 km.

2. GCF to PCF, starting from a GCF with $R = 1$ km and arriving to a PCF with $R = 2$ km. The GCF initial phase angle is 125 degree and $\rho_{y,0} = 0$.

3. GCF to ATF, starting from a GCF with $R = 1$ km and $\rho_{y,0} = 0$ and arriving to an ATF with $\rho_{y,0} = -0.3$ km. The GCF initial phase angle is 90 degree.

When dealing with a final condition given by a GCF or a PCF, an important difference with the cases faced in Sec. 9.6.1 and 9.6.2 is introduced. In fact, in these

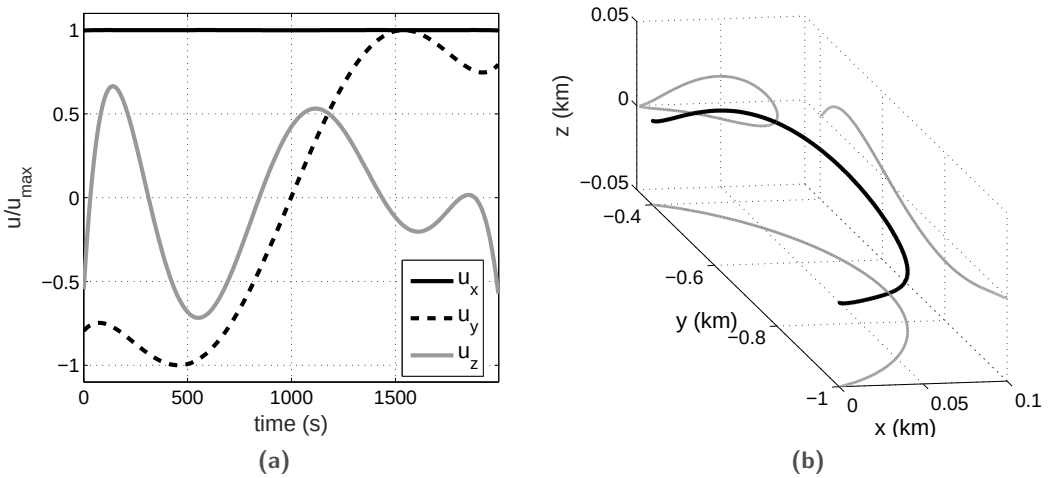


Fig. 9.9: Control policy (a) and trajectory (b) for the ATF to ATF maneuver.

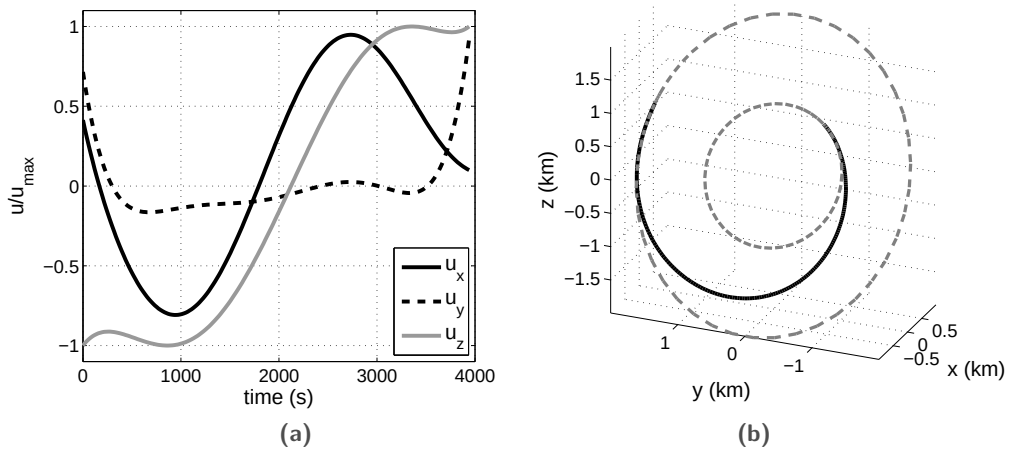


Fig. 9.10: Control policy (a) and trajectory (b) for the GCF to PCF maneuver.

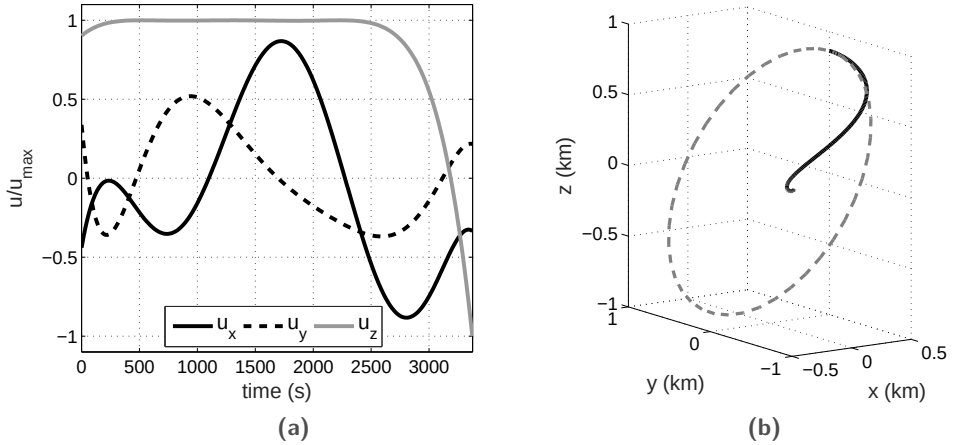


Fig. 9.11: Control policy (a) and trajectory (b) for the GCF to ATF maneuver.

cases the final condition does not corresponds to a point but to an infinite set of points satisfying Eq. (9.38) and parametrized by the initial phase angle α_0. Accordingly, the minimum time t_f depends on α_0 , and the DE individual in Eq. (9.30) must be modified to consider α_0. Hence, using the basic B-spline approximation, the DE individual is

$$\boldsymbol{X} = [\boldsymbol{a}_x, \boldsymbol{a}_y, \boldsymbol{a}_z, \alpha_0,\ t_f]\,. \tag{9.39}$$

The results obtained with the proposed DE-based planner using the basic B-spline approximation are reported in Fig. 9.9, 9.10 and 9.11. The maneuver times required for the ATF to ATF, GCF to PCF and GCF to ATF maneuvers are 1995 seconds, 3941 seconds and 3374 seconds, respectively. All the characteristics already pointed out in the previous sections apply to these cases. It is noteworthy that, for all the three reported maneuvers, at least one component of the external control is

quite completely extremal. For the ATF to ATF maneuver, u_x is constantly equal to $+1$ while u_y basically switches from -1 to $+1$ (note that, when the maneuver is performed in the opposite direction, i.e. going from -1 to -0.4 along the y axis, u_x and u_y are exactly the same with opposite sign, so $u_x = -1 \; \forall t$ and u_y basically switches from $+1$ to -1). For the GCF to PCF maneuver, u_z switches from -1 to $+1$. Analogously, for the GCF to ATF maneuver, u_z is equal to $+1$ for almost the entire duration of the maneuver, showing a little negative bang at the end. The presence of extremal controls is a good index of the quasi-optimality of the reported results, as this fact agrees with the theoretical analysis described in Sec. 9.2.

9.7 Endnotes

A novel algorithm for planning near time-optimal maneuvers has been proposed. The planner is based on the Differential Evolution algorithm and implements an inverse dynamics approach. Accordingly, the optimization variables are the parameters which describe the kinematic history of the deputy satellite. A comparison between the B-spline and the Chebyshev approximation methods has been performed, and the numerical results show that the former outperforms the latter. A Monte Carlo analysis has been carried out to demonstrate the stability and the efficiency of the proposed planner when using the B-spline approximation. Near time-optimal maneuvers are planned requiring less than two thousands iteration, on average, and a good approximation of a bang-bang control policy is attained. An improved B-spline approximation scheme has also been tested, demonstrating that the optimality of the trajectory may be improved at the cost of higher computational efforts. Moreover, the numerical results of the proposed planner have been compared with those obtained using a Particle Swarm optimizer. Results from Differential Evolution and from Particle Swarm are consistent with each other and, generally, the Differential Evolution outperforms the Particle Swarm.

As a conclusion, the proposed planner based on Differential Evolution and using B-spline to approximate the deputy trajectories may be successfully used to obtain feasible maneuvers while representing a respectable compromise between optimality and computational effort.

Conclusion

The thesis has focused on planning optimal maneuvers using metaheuristic methods. The Particle Swarm Optimization has been mainly used, but the Differential Evolution has been introduced in the final chapter to compare the performances of different metaheuristic algorithms.

The main contribution of this work has been the development of the Inverse-dynamics Particle Swarm Optimization, which is an optimal control algorithm based on a differential flatness implementation and the particle swarm intelligence to solve the nonlinear parameter optimization problem. The test cases reported in the thesis, concerning attitude maneuvers and satellite formation reconfiguration maneuvers, have demonstrated that the proposed algorithm may be successfully used to plan near-optimal maneuvers. For attitude maneuvers, the algorithm requires a very low computational effort. On the other hand, for satellite formation reconfiguration maneuvers, the algorithm has been used to demonstrate the feasibility of perturbation-based maneuvers, and the completely nonlinear spacecraft dynamics have been considered. Accordingly, a higher computational effort is required with respect to the planning of attitude maneuvers. In the thesis it has been shown that nonconvexity issues related to the differential flatness implementation may be easily overcome by using the particle swarm optimizer.

The particle swarm intelligence has been also employed to cope with direct dynamics formulations of optimal control problems. When the necessary optimality conditions are imposed a-priori, an extremal control (usually known as bang-bang policy) is searched for. Hence, the particle swarm may be used to find the optimal sequence of bangs and switches to plan minimum-time maneuvers.

To conclude, optimal control problems can be successfully solved by means of swarm intelligence. Several approaches may be used to find the optimal control policy. Depending on the characteristics of the problem and the required accuracy of the solution, inverse or direct dynamics approaches may be used. The results shown in this thesis pave the way to new numerical approaches that can be employed as alternative methods to estalished collocation and pseudospectral algorithms.

Vita

Dario Spiller was born in Rome, Italy, on March 16, 1988. After finishing high-school in 2007, he received his B.Sc. degree in Aerospace Engineering at Sapienza, University of Rome, in 2010. As his interest in Space Engineering grew, he made the decision to study Astronautical Engineering at Sapienza, University of Rome, and obtained his M.Sc. degree in 2013. He has worked as a temporary research fellow for two years working on projects for OHB/CGS and Finmeccanica Leonardo (previously Selex ES). In 2014 he entered the Ph.D. program in Mechanical and Aerospace Engineering working on the development of a novel optimal control numerical technique. From 2014 to 2015 he has been involved in the ESA project "Robust high rate determination algorithms based on star sensing" carried out by a team Sapienza-Selex ES. From August 2016 to December 2016 he spent 5 months at the Pennsylvania State University collaborating with Prof. R. Melton on the development of the optimization algorithm IPSO. In 2017 he won the Mobility Grant for Ph.D. students offered by Sapienza and worked at University on Florida with Prof. R. Bevilacqua from April to September. Here he studied advanced analytical techniques for relative motion maneuvering.

Bibliography

[1]D. Spiller, L. Ansalone, and F. Curti. “Particle Swarm Optimization for Time-Optimal Spacecraft Reorientation with Keep-Out Cones”. In: *Journal of Guidance, Control, and Dynamics* 39.2 (2016), pp. 312–325. DOI: 10.2514/1.G001228 (cit. on pp. 10, 17, 45, 72, 82, 84, 91, 93, 102, 104, 151, 154, 226).

[2]D. Spiller, F. Curti, and L. Ansalone. “Inverse Dynamics Particle Swarm Optimization for Spacecraft Minimum-Time Maneuvers with Constraints”. In: *23rd Conference of the Italian Association of Aeronautics and Astronautics* (2015) (cit. on pp. 10, 17, 72, 75, 91, 120, 226, 240).

[3]D. Spiller, F. Curti, and R. G. Melton. “Inverse Dynamics Particle Swarm Optimization Applied to Constrained Minimum-Time Maneuvers Using Reaction Wheels”. In: *67th International Astronautical Congress, Guadalajara, Mexico* (2016) (cit. on pp. 10, 17, 72, 75, 91, 127, 240).

[4]D. Spiller, F. Curti, and Ansalone. “Inverse-Dynamics Particle Swarm Optimization for Spacecraft Minimum-Time Slew Maneuvers with Constraints”. In: *Aerotecnica Missili & Spazio, The Journal of Aerospace Science, Technology and Systems* 96.3 (2017), pp. 111–123. DOI: 10.19249/ams.v96i3.302 (cit. on pp. 10, 72, 91).

[5]D.-Z. Du, P. M. Pardalos, and W. Wu. “History of optimization”. In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Boston, MA: Springer US, 2009, pp. 1538–1542. ISBN: 978-0-387-74759-0. DOI: 10.1007/978-0-387-74759-0_268 (cit. on p. 16).

[6]S. Kiranyaz, T. Ince, and M. Gabbouj. *Multidimensional Particle Swarm Optimization for Machine Learning and Pattern Recognition*. Adaptation, Learning, and Optimization. Springer Berlin Heidelberg, 2013. ISBN: 9783642378461 (cit. on p. 16).

[7]X. S. Yang. *Engineering Optimization - An Introduction with Metaheuristic Applications*. Ed. by Wiley. John Wiley & Sons, New Jersey, 2010. ISBN: 978-0-470-58246-6 (cit. on pp. 16, 28, 29).

[8]D. Spiller, F. Curti, and C. Circi. “Minimum-Time Reconfiguration Maneuvers of Satellite Formations Using Perturbation Forces”. In: *Journal of Guidance, Control and Dynamics* 40.5 (2017), pp. 1130–1143. DOI: 10.2514/1.G002382 (cit. on pp. 17, 72, 75, 82, 84, 179, 187, 189, 203–205, 209, 220, 221).

[9]D. Spiller and F. Curti. “Inverse Dynamics Particle Swarm Optimization for Nanosatellites Rendezvous via Differential Drag”. In: Rome, Italy, 2015 (cit. on pp. 17, 72, 75).

[10]D. G. Hull. “Conversion of optimal control problems into parameter optimization problems”. In: *Journal of Guidance, Control, and Dynamics* 20.1 (1997), pp. 57–60. DOI: 10.2514/2.4033 (cit. on pp. 19, 42).

[11]J. T. Betts. “Survey of numerical methods for trajectory optimization”. In: *Journal of Guidance control and dynamics* 21.2 (1998), pp. 193–207. DOI: 10.2514/2.4231 (cit. on pp. 19, 42, 224).

[12]S. S. Rao. *Engineering Optimization: Theory and Practice*. Wiley, 2009. ISBN: 9780470183526 (cit. on pp. 19–23, 27).

[13]G. Dantzig. *Linear Programming and Extensions*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, 2016. ISBN: 9781400884179 (cit. on p. 19).

[14]P. E. Gill, W. Murray, and M. A. Saunders. “SNOPT: An SQP algorithm for large-scale constrained optimization”. In: *SIAM review* 47.1 (2005), pp. 99–131. DOI: 10.1137/S1052623499350013 (cit. on p. 24).

[15]A. S. Nemirovski and M. J. Todd. “Interior-point methods for optimization”. In: *Acta Numerica* 17 (2008), pp. 191–234. DOI: 10.1017/S0962492904 (cit. on p. 27).

[16]A. Wächter and L. T. Biegler. “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical programming* 106.1 (2006), pp. 25–57. DOI: 10.1007/s10107-004-0559-y (cit. on p. 27).

[17]S. Voss. “Metaheuristics”. In: *Encyclopedia of Optimization*. Ed. by C. A. Floudas and P. M. Pardalos. Boston, MA: Springer US, 2009, pp. 1538–1542. ISBN: 978-0-387-74759-0. DOI: 10.1007/978-0-387-74759-0_367 (cit. on pp. 28, 29).

[18]H. E. Romeijn. “Random search methods”. In: *Encyclopedia of Optimization*. Ed. by Christodoulos A. Floudas and Panos M. Pardalos. Boston, MA: Springer US, 2009, pp. 3245–3251. ISBN: 978-0-387-74759-0. DOI: 10.1007/978-0-387-74759-0_556 (cit. on p. 29).

[19]J. Pearl. *Heuristics: intelligent search strategies for computer problem solving*. The Addison-Wesley series in artificial intelligence. Addison-Wesley Pub. Co, 1984. ISBN: 9780201055948 (cit. on p. 29).

[20]I. H. Osman and J. P. Kelly. *Meta-Heuristics: Theory and Applications*. Springer US, 2012. ISBN: 9781461313618 (cit. on p. 29).

[21]S. Voß, S. Martello, I. H. Osman, and C. Roucairol. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Springer US, 2012. ISBN: 9781461557753 (cit. on p. 29).

[22]I. Boussaïd, J. Lepagnot, and P. Siarry. “A survey on optimization metaheuristics”. In: *Information Sciences* 237 (2013), pp. 82–117. DOI: 10.1016/j.ins.2013.02.041 (cit. on pp. 29–31).

[23]M. Črepinšek, S.-H. Liu, and M. Mernik. “Exploration and exploitation in evolutionary algorithms: a survey”. In: *ACM Computing Surveys (CSUR)* 45.3 (2013), p. 35. DOI: 10.1145/2480741.2480752 (cit. on p. 30).

[24]M. Mavrovouniotis, C. Li, and S. Yang. “A survey of swarm intelligence for dynamic optimization: algorithms and applications”. In: *Swarm and Evolutionary Computation* 33 (2017), pp. 1–17. DOI: 10.1016/j.swevo.2016.12.005 (cit. on p. 31).

[25]M. Athans and L. F. Peter. *Optimal control : an introduction to the theory and its applications*. Dover, 2007. ISBN: 0486453286 (cit. on pp. 35, 37, 40, 130, 151, 153, 223).

[26]D. E. Kirk. *Optimal Control Theory: An Introduction*. Dover Books on Electrical Engineering. Dover Publications, 2012. ISBN: 9780486135076 (cit. on pp. 35, 37, 158, 160).

[27]I. M. Ross. *A Primer on Pontryagin's Principle in Optimal Control: Second Edition*. Collegiate Publishers, 2015. ISBN: 9780984357116 (cit. on pp. 35, 47, 67, 86).

[28]R. F. Hartl, S. P. Sethi, and R. G. Vickson. "A survey of the maximum principles for optimal control problems with state constraints". In: *SIAM review* 37.2 (1995), pp. 181–218. DOI: 10.1137/1037043 (cit. on p. 35).

[29]L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. Interscience Publishers, John Wiley & Sons, 1965 (cit. on pp. 36, 37, 150, 158, 160, 223).

[30]"The Maximum Principal and the Solution of Two-Point Boundary Value Problems". In: *The Computation and Theory of Optimal Control*. Ed. by P. Dyer and S. R. McReynolds. Vol. 65. Elsevier Science, 1970. Chap. 8, pp. 213 –234. ISBN: 9780080955742. DOI: 10.1016/S0076-5392(08)63412-8 (cit. on p. 36).

[31]G. V. Smirnov. *Introduction to the Theory of Differential Inclusions (Graduate Studies in Mathematics, v. 41)*. Amer Mathematical Society, 2002. DOI: ISBN:0-08218-2977-7 (cit. on p. 40).

[32]M. Fliess, J. Lévine, P. Martin, and P. Rouchon. "Flatness and defect of non-linear systems: introductory theory and examples". In: *International Journal of Control* 61 (2007), pp. 1327–1361. DOI: 10.1080/00207179508921959 (cit. on pp. 40, 50, 148, 224).

[33]F. Fahroo and I. M. Ross. "Second look at approximating differential inclusions". In: *Journal of Guidance, Control, and Dynamics* 24.1 (2001), pp. 131–133. DOI: 10.2514/2.4686 (cit. on pp. 40, 49).

[34]I. M. Ross and F. Fahroo. "Issues in the real-time computation of optimal control". In: *Mathematical and computer modelling* 43.9 (2006), pp. 1172–1188. DOI: 10.1016/j.mcm.2005.05.021 (cit. on pp. 40, 41, 43, 145, 150).

[35]V. Azhmyakov and J. Raisch. "Convex Control Systems and Convex Optimal Control Problems With Constraints". In: *IEEE Transactions on Automatic Control* 53.4 (2008), pp. 993–998. DOI: 10.1109/TAC.2008.919848 (cit. on pp. 40, 144).

[36]B. S. Mordukhovich. "Optimal control of nonconvex differential inclusions". In: *Differential Equations, Chaos and Variational Problems*. Springer, 2007, pp. 285–303. DOI: 10.1007/978-3-7643-8482-123 (cit. on pp. 41, 47).

[37]B. Mordukhovich and D. Wang. "Optimal control of differential inclusions in infinite-dimensional spaces". In: *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*. Vol. 1. 2004, pp. 899–904. DOI: 10.1109/CDC.2004.1428799 (cit. on pp. 41, 47).

[38]Y. Mao, M. Szmuk, and B. Açikmeşe. “Successive convexification of non-convex optimal control problems and its convergence properties”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. 2016, pp. 3636–3641. DOI: 10.1109/CDC.2016.7798816 (cit. on p. 41).

[39]X. Liu and P. Lu. “Solving nonconvex optimal control problems by convex optimization”. In: *Journal of Guidance, Control, and Dynamics* 37.3 (2014), pp. 750–765. DOI: 10.2514/1.62110 (cit. on p. 41).

[40]I. M. Ross and F. Fahroo. “Legendre pseudospectral approximations of optimal control problems”. In: *New trends in nonlinear dynamics and control and their applications*. Springer, 2003, pp. 327–342. DOI: doi:10.1007/978-3-540-45056-6_21 (cit. on pp. 41, 45, 47).

[41]S. S. Rao. *Engineering Optimization: Theory and Practice: Fourth Edition*. John Wiley and Sons, June 2009. ISBN: 9780470183526. DOI: 10.1002/9780470549124 (cit. on p. 41).

[42]B. Açikmeşe and L. Blackmore. “Lossless convexification of a class of optimal control problems with non-convex control constraints”. In: *Automatica* 47.2 (2011), pp. 341–347. DOI: 10.1016/j.automatica.2010.10.037 (cit. on p. 41).

[43]O. Von Stryk and R. Bulirsch. “Direct and indirect methods for trajectory optimization”. In: *Annals of operations research* 37.1 (1992), pp. 357–373. DOI: 10.1007/BF02071065 (cit. on p. 42).

[44]C. R. Hargraves and S. W. Paris. “Direct trajectory optimization using nonlinear programming and collocation”. In: *Journal of Guidance, Control, and Dynamics* 10.4 (1987), pp. 338–342. DOI: 10.2514/3.20223 (cit. on pp. 42, 47).

[45]J. T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Second. Society for Industrial and Applied Mathematics, 2010. ISBN: 0-89871-488-5. DOI: 10.1137/1.9780898718577 (cit. on pp. 42, 74, 224).

[46]J. Z. Ben-Asher. *Optimal control theory with aerospace applications*. American Institute of Aeronautics and Astronautics, 2010. ISBN: 978 1 60086 732 3. DOI: 10.2514/4.867347 (cit. on p. 42).

[47]I. M. Ross and F. Fahroo. “A perspective on methods for trajectory optimization”. In: *Proceedings of the AIAA/AAS Astrodynamics Conference, Monterey, CA*. 2002. DOI: 10.2514/6.2002-4727 (cit. on pp. 42, 50, 145, 147).

[48]T. Veeraklaew and S. K. Agrawal. “New Computational Framework for Trajectory Optimization of Higher-Order Dynamic Systems”. In: *Journal of Guidance Control and Dynamics* 24 (2 Mar. 2001). DOI: 10.2514/2.4733 (cit. on p. 43).

[49]A. V. Rao. “A survey of numerical methods for optimal control”. In: *Advances in the Astronautical Sciences* 135.1 (2009), pp. 497–528 (cit. on p. 47).

[50]G. Elnagar, M. A. Kazemi, and M. Razzaghi. “The pseudospectral Legendre method for discretizing optimal control problems”. In: *IEEE transactions on Automatic Control* 40.10 (1995), pp. 1793–1796. DOI: 10.1109/9.467672 (cit. on p. 47).

[51]C. Canuto, M. Y. Hussaini, A. Quarteroni, and J. Z. Thomas A. *Spectral Methods in Fluid Dynamics*. Scientific Computation. Springer Berlin Heidelberg, 2012. ISBN: 9783642841088 (cit. on p. 47).

[52]G. T. Huntington and A. V. Rao. “Comparison of global and local collocation methods for optimal control”. In: *Journal of guidance, control, and dynamics* 31.2 (2008), p. 432 (cit. on pp. 47, 202).

[53]D. Benson. “A Gauss pseudospectral transcription for optimal control”. PhD thesis. Massachusetts Institute of Technology, 2005 (cit. on p. 47).

[54]G. T. Huntington. *Advancement and analysis of a Gauss pseudospectral transcription for optimal control problems*. 2007 (cit. on p. 47).

[55]C. L. Darby, W. W. Hager, and A. V. Rao. “An hp-adaptive pseudospectral method for solving optimal control problems”. In: *Optimal Control Applications and Methods* 32.4 (2011), pp. 476–502. DOI: 10.1002/oca.957 (cit. on p. 47).

[56]D. Garg. “Advances in global pseudospectral methods for optimal control”. PhD thesis. University of Florida, 2011 (cit. on p. 47).

[57]M. A. Patterson and A. V. Rao. “GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming”. In: *ACM Transactions on Mathematical Software* 41.1 (2014), pp. 1–37. DOI: 10.1145/2558904 (cit. on pp. 47, 102, 109, 137, 153, 175, 241).

[58]G. A. Boyarko, M. Romano, and O. A. Yakimenko. “Time-Optimal Reorientation of a Spacecraft Using an Inverse Dynamics Optimization Method”. In: *Journal of Guidance, Control, and Dynamics* 34 (4) (2011), pp. 1197–1208. DOI: 10.2514/1.49449 (cit. on p. 47).

[59]J. Ventura, M. Ciarcia, M. Romano, and U. Walter. “An Inverse Dynamics-Based Trajectory Planner for Autonomous Docking to a Tumbling Target”. In: *Proceedings of the AIAA Guidance, Navigation and Control Conference* (2016). DOI: 10.2514/6.2016-0876 (cit. on p. 47).

[60]H. Seywald. “Trajectory optimization based on differential inclusion”. In: *Journal of Guidance Control and Dynamics* 17.3 (1994), pp. 480–487. DOI: 10.2514/3.21224 (cit. on p. 47).

[61]S. Raczynski. “Some remarks on nonconvex optimal control”. In: *Journal of mathematical analysis and applications* 118.1 (1986), pp. 24–37. DOI: 10.1016/0022-247X(86)90287-8 (cit. on p. 47).

[62]B. S. Mordukhovich and H. J. Sussmann. *Nonsmooth Analysis and Geometric Methods in Deterministic Optimal Control*. The IMA Volumes in Mathematics and its Applications. Springer, 1996, pp. 153–202. ISBN: ISBN-13: 978-1-4613-8491-5 (cit. on p. 47).

[63]R. R. Kumar and H. Seywald. “Should controls be eliminated while solving optimal control problems via direct methods?” In: *Journal of Guidance Control and Dynamics* 19 (1996), pp. 418–423. DOI: 10.2514/3.21634 (cit. on p. 49).

[64]B. A. Conway and K. M. Larson. “Collocation versus differential inclusion in direct optimization”. In: *Journal of Guidance Control and Dynamics* 21 (1998), pp. 780–785. DOI: 10.2514/2.4306 (cit. on p. 49).

[65]C. Louembet. “Design of Algorithms for Satellite Slew Manoeuver by Flatness and Collocation”. In: *American Control Conference* (2007), pp. 3168–3173. DOI: 10.1109/ACC.2007.4282459 (cit. on p. 50).

[66]J. Kennedy and R. Eberhart. “Particle Swarm Optimization”. In: *Proceedings of the IEEE International Conference on Neural Networks* 4 (1995), pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968 (cit. on pp. 56, 57, 59, 60, 72).

[67]J. F. Kennedy, J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm Intelligence*. Evolutionary Computation Series. Morgan Kaufmann Publishers, 2001. ISBN: 9781558605954 (cit. on pp. 56, 72, 242).

[68]M. Clerc. *Particle Swarm Optimization*. ISTE. Wiley, 2013. ISBN: 9781118613979 (cit. on pp. 56, 64).

[69]K. E. Parsopoulos. *Particle Swarm Optimization and Intelligence: Advances and Applications: Advances and Applications*. Advances in Computational Intelligence and Robotics: Information Science Reference, 2010. ISBN: 9781615206674 (cit. on pp. 56, 65).

[70]Y. Shi and R. C. Eberhart. “A Modified Particle Swarm Optimizer”. In: *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. Washington, DC, USA: IEEE Computer Society, May 1998, pp. 69–73. DOI: 10.1109/ICEC.1998.699146 (cit. on pp. 58, 59).

[71]Y. Shi and R. C. Eberhart. “Empirical study of particle swarm optimization”. In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. Vol. 3. IEEE. Washington, DC, USA, 1999, pp. 1945–1950. DOI: 10.1109/CEC.1999.785511 (cit. on p. 59).

[72]M. M. Millonas. “Swarms, phase transitions, and collective intelligence”. In: *Santa Fe Institute Studies in the Sciences of Complexity*. Vol. 17. ADDISON-WESLEY PUBLISHING CO. 1994, pp. 417–417 (cit. on p. 61).

[73]R. Eberhart and J. Kennedy. “A new optimizer using particle swarm theory”. In: *Micro Machine and Human Science, 1995. MHS’95., Proceedings of the Sixth International Symposium on*. IEEE. 1995, pp. 39–43. DOI: 10.1109/MHS.1995.494215 (cit. on p. 62).

[74]J. Kennedy. “The particle swarm: social adaptation of knowledge”. In: *Evolutionary Computation, 1997., IEEE International Conference on*. IEEE. 1997, pp. 303–308. DOI: 10.1109/ICEC.1997.592326 (cit. on p. 63).

[75]K.E. Parsopoulos and M.N. Vrahatis. “Parameter selection and adaptation in Unified Particle Swarm Optimization”. In: *Mathematical and Computer Modelling* 46 (2007), pp. 198–213. DOI: 10.1016/j.mcm.2006.12.019 (cit. on pp. 63, 64).

[76]C. De Boor. “On Calculating with B-splines”. In: *Journal of Approximation Theory* 6 (1972), pp. 50–62. DOI: 10.1016/0021-9045(72)90080-9 (cit. on pp. 72, 74, 231).

[77]C. De Boor. “Splines as linear combinations of B-splines. A Survey”. In: *Journal of Approximation Theory* (1986). DOI: 10.1.1.34.8204 (cit. on pp. 72, 74, 231).

[78]M. G. Cox. “Practical spline approximation”. In: *Topics in Numerical Analysis*. Springer, 1982, pp. 79–112. DOI: 10.1007/BFb0063201 (cit. on pp. 74, 231).

[79]A. Saxena and B. Sahay. *Computer Aided Engineering Design*. Springer Netherlands, 2007. ISBN: 9781402038716 (cit. on p. 74).

[80]K. D. Bilimoria and B. Wie. “Time-optimal three-axis reorientation of rigid spacecraft”. In: *Journal of Guidance, Control, and Dynamics* 16 (3) (1993), pp. 446–452. DOI: 10.2514/3.21030 (cit. on pp. 92, 94, 171, 175).

[81]F. Li and P. M. Bainum. “Numerical approach for solving rigid spacecraft minimum time attitude maneuvers”. In: *J. Guidance, Control, Dynamics* 13.1 (1990), pp. 38–45. DOI: 10.2514/3.20515 (cit. on p. 92).

[82]S. L. Scrivener and R. C. Thompson. “Survey of time-optimal attitude maneuvers”. In: *Journal of Guidance, Control, and Dynamics* 17.2 (1994), pp. 225–233. DOI: 10.2514/3.21187 (cit. on p. 92).

[83]R. M. Byers and S. R. Vadali. “Quasi-closed-form solution to the time-optimal rigid spacecraft reorientation problem”. In: *Journal of Guidance, Control, and Dynamics* 16.3 (1993), pp. 453–461. DOI: 10.2514/3.21031 (cit. on p. 92).

[84]X. Bai and J. L. Junkins. “New Results for Time-Optimal Three-Axis Reorientation of a Rigid Spacecraft”. In: *Journal of Guidance, Control, and Dynamics* 32 (4) (2009), pp. 1071–1076. DOI: 10.2514/1.43097 (cit. on pp. 92, 175).

[85]H. Shen and R. Tsiotras. “Time-optimal control of axisymmetric rigid spacecraft using two controls”. In: *Journal of Guidance Control and Dynamics* 22 (1999), pp. 682–694. DOI: 10.2514/2.4436 (cit. on p. 92).

[86]A. Fleming, P. Sekhavat, and I. M. Ross. “Minimum-time reorientation of a rigid body”. In: *Journal of guidance, control, and dynamics* 33.1 (2010), p. 160. DOI: 10.2514/1.43549 (cit. on p. 92).

[87]S.-W. Liu and T. Singh. “Fuel/time optimal control of spacecraft maneuvers”. In: *Journal of guidance, control, and dynamics* 20.2 (1997), pp. 394–397. DOI: 10.2514/2.4053 (cit. on p. 92).

[88]J. Li. “Time-optimal three-axis reorientation of asymmetric rigid spacecraft via homotopic approach”. In: *Advances in space research* 57.10 (2016), pp. 2204–2217. DOI: 10.1016/j.asr.2016.02.016 (cit. on p. 92).

[89]J. Li and X.-N. Xi. “Time-optimal reorientation of the rigid spacecraft using a pseudospectral method integrated homotopic approach”. In: *Optimal Control Applications and Methods* 36.6 (2015), pp. 889–918. DOI: 10.1002/oca.2145 (cit. on p. 92).

[90]B. M. Yutko and R. G. Melton. “Optimizing spacecraft reorientation maneuvers using a pseudospectral method”. In: *Journal of Aerospace Engineering, Sciences and Applications* 2.1 (2010), pp. 1–14 (cit. on pp. 92, 95).

[91]C. R. McInnes. “Large angle slew maneuvers with autonomous sun vector avoidance”. In: *Journal of Guidance, Control, and Dynamics* 17 (4) (1994), pp. 875–877. DOI: 10.2514/3.21283 (cit. on p. 92).

[92]J. D. Koenig. “A novel attitude guidance algorithm for exclusion zone avoidance”. In: *Aerospace conference, 2009 IEEE*. IEEE. 2009, pp. 1–10. DOI: 10.1109/AERO.2009.4839538 (cit. on p. 92).

[93]H. B. Hablani. “Attitude Commands Avoiding Bright Objects and Maintaining Communication with Ground Station”. In: *Journal of Guidance, Control, and Dynamics* 22 (6) (1999), pp. 759–767. DOI: 10.2514/2.4469 (cit. on p. 92).

[94]F. Boldrini, D. Procopio, S. P. Airy, and L. Giulicchi. “Miniaturised Star Tracker (AASTR) ready to fly”. In: *Proceedings of the 4S Symposium: Small Satellites, Systems and Services (ESA SP-571)* (2004) (cit. on p. 92).

[95]U. Schmidt, T. Fiksel, A. Kwiatkowski, et al. “Autonomous star sensor ASTRO APS: flight experience on Alphasat”. In: *CEAS Space Journal* (2015), pp. 1–10. DOI: 10.1007/s12567-014-0071-z (cit. on p. 92).

[96]R. G. Melton. “Numerical analysis of constrained, time-optimal satellite reorientation”. In: *Mathematical Problems in Engineering* 2012 (2012), pp. 1–19. DOI: 10.1155/2012/769376 (cit. on p. 92).

[97]E. Frazzoli, M. A. Dahleh, E. Feron, and R. Kornfeld. “A randomized attitude slew planning algorithm for autonomous spacecraft”. In: *AIAA Guidance, Navigation, and Control Conference and Exhibit, Montreal, Canada*. Montreal, Quebec, Canada, 2001 (cit. on p. 92).

[98]U. Lee and M. Mesbahi. “Spacecraft Reorientation in Presence of Attitude Constraints via Logarithmic Barrier Potentials”. In: *Proceedings of the American Control Conference* (2011), pp. 450 –455. DOI: 10.1109/ACC.2011.5991284 (cit. on p. 93).

[99]T. Lee, M. Leok, and N. H. McClamroch. “Time optimal attitude control for a rigid body”. In: *American Control Conference, 2008*. IEEE. 2008, pp. 5210–5215. DOI: 10.1109/ACC.2008.4587322 (cit. on p. 93).

[100]P. Cui, W. Zhong, and H. Cui. “Onboard Spacecraft Slew-Planning by Heuristic State-Space Search and Optimization”. In: *Proceedings of the 2007 International Conference on Mechatronics and Automation* (2007), pp. 2115–2119. DOI: 10.1109/ICMA.2007.4303878 (cit. on pp. 93, 96, 224).

[101]L.-C. Lai, C.-C. Yang, and C.-J. Wu. “Time-optimal maneuvering control of a rigid spacecraft”. In: *Acta Astronautica* 60.10 (2007), pp. 791–800. DOI: 10.1016/j.actaastro.2006.09.039 (cit. on p. 93).

[102]R. G. Melton. “Hybrid methods for determining time-optimal, constrained spacecraft reorientation maneuvers”. In: *Acta Astronautica* 94 (2014), pp. 294–301. DOI: 10.1016/j.actaastro.2013.05.007 (cit. on pp. 93, 119, 126).

[103]R. G. Melton. “Maximum-likelihood estimation optimizer for constrained, time-optimal satellite reorientation”. In: *Acta Astronautica* 103 (2014), pp. 185–192. DOI: 10.1016/j.actaastro.2014.06.032 (cit. on p. 93).

[104]R. P. Kornfeld. *On-board autonomous attitude maneuver planning for planetary spacecraft using genetic algorithms*. Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2003 (cit. on p. 93).

[105]Y. S. X. Shijie. “Spacecraft attitude maneuver planning based on particle swarm optimization”. In: *Journal of Beijing University of Aeronautics and Astronautics* 1 (2010), p. 013 (cit. on p. 93).

[106]D. J. Showalter and J. T. Black. “Responsive theater maneuvers via particle swarm optimization”. In: *Journal of Spacecraft and Rockets* 51.6 (2014), pp. 1976–1985. DOI: 10.2514/1.A32989 (cit. on p. 93).

[107]M. Pontani and B. A. Conway. “Particle Swarm Optimization Applied to Space Trajectories”. In: *Journal of Guidance, Control, and Dynamics* 33 (5) (2010), pp. 1429–1441. DOI: 10.2514/1.48475 (cit. on p. 93).

[108]P. Ghosh and B. A. Conway. “Numerical trajectory optimization with swarm intelligence and dynamic assignment of solution structure”. In: *Journal of Guidance, Control and Dynamics* 35.4 (2012), pp. 1178–1191. DOI: 10.2514/1.55594 (cit. on pp. 93, 159, 164, 166, 168, 169).

[109]W. Xu, C. Li, X. Wang, et al. “Study on non-holonomic cartesian path planning of a free-floating space robotic system”. In: *Advanced Robotics* 23.1-2 (2009), pp. 113–143. DOI: 10.1163/156855308X392708 (cit. on p. 93).

[110]P. Huang, G. Liu, J. Yuan, and Y. Xu. “Multi-objective optimal trajectory planning of space robot using particle swarm optimization”. In: *Advances in Neural Networks-ISNN 2008* (2008), pp. 171–179. DOI: 10.1007/978-3-540-87734-9-20 (cit. on p. 93).

[111]A. Rahimi, K. D. Kumar, and H. Alighanbari. “Particle swarm optimization applied to spacecraft reentry trajectory”. In: *Journal of Guidance, Control, and Dynamics* 36.1 (2013), pp. 307–310. DOI: 10.2514/1.56387 (cit. on p. 93).

[112]S.-M. Chen and Y.-F. Dong. “Satellite Attitude Tracking Controller Optimization Based on Particle Swarm Optimization”. In: *Procedia Engineering* 15 (2011), pp. 526–530. DOI: 10.1016/j.proeng.2011.08.100 (cit. on p. 93).

[113]N. Xia, D. Han, G. Zhang, J. Jiang, and K. Vu. “Study on attitude determination based on discrete particle swarm optimization”. In: *Science China Technological Sciences* 53.12 (2010), pp. 3397–3403. DOI: 10.1007/s11431-010-4148-4 (cit. on p. 93).

[114]M. D. Shuster. “A Survey of Attitude Representations”. In: *The Journal of the Astronautical Sciences* 41 (4) (1993), pp. 439–517 (cit. on pp. 96, 102, 184, 199).

[115]S. Vadali and J. Junkins. “Spacecraft large angle rotational maneuvers with optimal momentum transfer”. In: *Astrodynamics Conference*. Ed. by ARC AIAA. San Diego, CA, U.S.A., 1982. DOI: 10.2514/6.1982-1469 (cit. on p. 127).

[116]W. Steyn. “Near-minimum-time eigenaxis rotation maneuvers using reaction wheels”. In: *Journal of Guidance, Control, and Dynamics* 18 (5) (1995), pp. 1184–1189. DOI: 10.2514/3.21523 (cit. on p. 127).

[117]Y. W. Jan and J. C. Chiou. “Minimum-time spacecraft maneuver using sliding-mode control”. In: *Acta Astronautica* 54.1 (2004), pp. 69–75. DOI: 10.1016/S0094-5765(03)00194-2 (cit. on p. 127).

[118]H. Zhou, D. Wang, B. Wu, and E. K. Poh. “Time-optimal reorientation for rigid satellite with reaction wheels”. In: *International Journal of Control* 85.10 (2012), pp. 1452–1463. DOI: 10.1080/00207179.2012.688873 (cit. on p. 127).

[119]F. L. Markley and J. L. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. Ed. by Space Technology Library. Springer, NY, 2014. ISBN: 978-1-4939-0801-1 (cit. on p. 128).

[120]J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons Ltd, England, 2008. ISBN: 978-0-470-72335-7 (cit. on p. 132).

[121]M. Ross and F. Fahroo. “Issues in the real-time computation of optimal control”. In: *Mathematical and Computer Modelling* 43 (2005), pp. 1172–1188. DOI: 10.1016/j.mcm.2005.05.021 (cit. on pp. 138, 220).

[122]D. Spiller, R. G. Melton, and F. Curti. “Inverse Dynamics Particle Swarm Optimizatino Applied to Bolza Problems”. In: Stevenson, Washington, US., 2017 (cit. on pp. 143, 226).

[123]X. Bai and J. L. Junkins. “New results for time-optimal three-axis reorientation of a rigid spacecraft”. In: *Journal of guidance, control, and dynamics* 32.4 (2009), p. 1071. DOI: 10.2514/1.43097 (cit. on p. 151).

[124]D. Spiller, C. Christian, and F. Curti. “Particle Swarm with Domain Partition and Control Assignment for Time Optimal Maneuvers”. In: *Journal of Guidance, Control, and Dynamics* (2017). Accepted for publication. In print. DOI: 10.2514/1.G002980 (cit. on p. 157).

[125]L. G. Van Willigenburg and R. P. H. Loop. “Computation of time-optimal controls applied to rigid manipulators with friction”. In: *International journal of control* 54.5 (1991), pp. 1097–1117. DOI: 10.1080/00207179108934200 (cit. on p. 159).

[126]A. Y. Lee. “Solving constrained minimum-time robot problems using the sequential gradient restoration algorithm”. In: *Optimal Control Applications and Methods* 13.2 (1992), pp. 145–154. DOI: 10.1002/oca.4660130205 (cit. on pp. 159, 166, 168).

[127]D. Spiller and K. Basu. “Optimal Passive Formation Reconfiguration using Attitude Control and Perturbing Forces”. In: Boulder, Colarado, US., 2017 (cit. on p. 179).

[128]C. Lambert, A. Ng, Y. Nakamura, and H. Horiguchi. “Intersatellite Separation Mechanism for the JC2Sat Formation-Flying Missions”. In: *Journal of Guidance, Control and Dynamics* 48.4 (2011), pp. 654–663. DOI: 10.2514/1.51896 (cit. on p. 180).

[129]D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen. “A Survey of Spacecraft Formation Flying Guidance and Control (Part I): Guidance”. In: *Chapter of The Path to Autonomous Robots* (2003). DOI: 10.1109/ACC.2003.1239845 (cit. on p. 180).

[130]D. P. Scharf, F. Y. Hadaegh, and S. R. Ploen. “A Survey of Spacecraft Formation Flying Guidance and Control (Part II): Control”. In: *Proceedings of the American Control Conference* (2004). DOI: 10.1109/ACC.2004.182741 (cit. on p. 180).

[131]W. Clohessy and R. Wiltshire. “Terminal Guidance System for Satellite Rendezvous”. In: *Journal of the Aerospace Sciences* 27.9 (1960), pp. 653–678. DOI: 10.2514/8.8704 (cit. on pp. 180, 186, 220).

[132]K. Alfriend, S. R. Vadali, P. Gurfil, J. How, and L. Breger. *Spacecraft Formation Flying: Dynamics, Control and Navigation*. Elsevier Astrodynamics Series. Elsevier Science, 2009. ISBN: 9780080559650 (cit. on pp. 180, 184, 186, 189, 192, 220, 234, 242).

[133]C. J. Scott and D. B. Spencer. “Optimal reconfiguration of satellites in formation”. In: *Journal of Spacecraft and Rockets* 44.1 (2007), pp. 230–239. DOI: 10.2514/1.21443 (cit. on p. 180).

[134]M. Massari and F. Bernelli-Zazzera. “Optimization of low-thrust reconfiguration maneuvers for spacecraft flying in formation”. In: *Journal of Guidance, Control, and Dynamics* 32.5 (2009), pp. 1629–1638. DOI: 10.2514/1.37335 (cit. on p. 180).

[135]G. T. Huntington and A. V. Rao. “Optimal reconfiguration of spacecraft formations using the Gauss pseudospectral method”. In: *Journal of Guidance, Control, and Dynamics* 31.3 (2008), pp. 689–698. DOI: 10.2514/1.31083 (cit. on p. 180).

[136]A. Richards, T. Schouwenaars, J. P. How, and E. Feron. “Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming”. In: *Journal of Guidance, Control, and Dynamics* 25.4 (2002), pp. 755–764. DOI: 10.2514/2.4943 (cit. on p. 180).

[137]H. Huang and Y. Zhuang. “Optimal satellite formation reconfiguration using co-evolutionary particle swarm optimization in deep space”. In: *Acta Astronautica* 113 (2015), pp. 149–163. DOI: 10.1016/j.actaastro.2015.04.003 (cit. on p. 181).

[138]C. Sun, H. Duan, and Y. Shi. “Optimal satellite formation reconfiguration based on closed-loop brain storm optimization”. In: *IEEE Computational Intelligence Magazine* 8.4 (2013), pp. 39–51. DOI: 10.1109/MCI.2013.2279560 (cit. on p. 181).

[139]J. Tian, N. Cui, and R. Mu. “Optimal formation reconfiguration using genetic algorithms”. In: *Computer Modeling and Simulation, 2009. ICCMS’09. International Conference on*. IEEE. 2009, pp. 95–98. DOI: 10.1109/ICCMS.2009.54 (cit. on p. 181).

[140]Y. Yoshimura and S. Hokamoto. “Optimal formation reconfiguration of satellites with attitude constraints using thrusters”. In: *5th International Conference on Spacecraft Formation Flying Missions and Technologies*. 2013 (cit. on p. 181).

[141]G. S. Aoude, J. P. How, and I. M. Garcia. “Two-stage path planning approach for solving multiple spacecraft reconfiguration maneuvers”. In: *The Journal of the Astronautical Sciences* 56.4 (2008), pp. 515–544. DOI: 10.1007/BF03256564 (cit. on p. 181).

[142]C. L. Leonard, W. M. Hollister, and E. V. Bergmann. “Orbital formationkeeping with differential drag”. In: *Journal of Guidance, Control, and Dynamics* 12.1 (1989), pp. 108–113. DOI: 10.2514/3.20374 (cit. on p. 181).

[143]R. Bevilacqua and M. Romano. “Rendezvous Maneuvers of Multiple Spacecraft Using Differential Drag Under J2 Perturbation”. In: *Journal of Guidance, Control and Dynamics* 31.6 (2008), pp. 1595–1607. DOI: 10.2514/1.36362 (cit. on p. 181).

[144]R. Bevilacqua, J. S Hall, and M. Romano. “Multiple spacecraft rendezvous maneuvers by differential drag and low thrust engines”. In: *Celestial Mechanics and Dynamical Astronomy* 106.1 (2010), p. 69. DOI: 10.1007/s10569-009-9240-3 (cit. on p. 181).

[145]D. Pérez and R. Bevilacqua. “Differential drag spacecraft rendezvous using an adaptive Lyapunov control strategy”. In: *Acta Astronautica* 83 (2013), pp. 196–207. DOI: 10.1016/j.actaastro.2012.09.005 (cit. on p. 181).

[146]D. Pérez and R. Bevilacqua. “Lyapunov-based adaptive feedback for spacecraft planar relative maneuvering via differential drag”. In: *Journal of Guidance, Control, and Dynamics* 37.5 (2014), pp. 1678–1684. DOI: 10.2514/1.G000191 (cit. on p. 181).

[147]M. Horsley, S. Nikolaev, and A. Pertica. “Small satellite rendezvous using differential lift and drag”. In: *Journal of Guidance, Control, and Dynamics* 36.2 (2013), pp. 445–453. DOI: 10.2514/1.57327 (cit. on p. 181).

[148]T. Reid and A. K. Misra. “Formation flight of satellites in the presence of atmospheric drag”. In: *Journal of Aerospace Engineering* 3.1 (2011), p. 64. DOI: 10.7446/jaesa.0301.05 (cit. on p. 181).

[149]K. D. Kumar, A. K. Misra, S. Varma, T. Reid, and F. Bellefeuille. “Maintenance of satellite formations using environmental forces”. In: *Acta Astronautica* 102 (2014), pp. 341–354. DOI: 10.1016/j.actaastro.2014.05.001 (cit. on p. 181).

[150]S. Varmas and K.D. Kumar. “Multiple Satellite Formation Flying Using Differential Aerodynamic Drag”. In: *Journal of Spacecraft and Rockets* 49.2 (2012), pp. 325–336. DOI: 10.2514/1.52395 (cit. on p. 181).

[151]C. Circi. “Simple strategy for geostationary stationkeeping maneuvers using solar sail”. In: *Journal of Guidance, Control, and Dynamics* 28.2 (2005), pp. 249–253. DOI: 10.2514/1.6797 (cit. on p. 181).

[152]K. Shahid and K. D. Kumar. “Formation control at the sun-earth L2 libration point using solar radiation pressure”. In: *Journal of Spacecraft and Rockets* 47.4 (2010), pp. 614–626. DOI: 10.2514/1.47342 (cit. on p. 181).

[153]T. Williams and Z.-S. Wang. “Uses of solar radiation pressure for satellite formation flight”. In: *International Journal of Robust and Nonlinear Control* 12.2-3 (2002), pp. 163–183. DOI: 10.1002/rnc.681 (cit. on p. 181).

[154]K. Shahid and K. D. Kumar. “Multiple spacecraft formation reconfiguration using solar radiation pressure”. In: *Acta Astronautica* 103 (2014), pp. 269–281. DOI: 10.1016/j.actaastro.2014.05.021 (cit. on p. 181).

[155]Y.-G. Hou, M.-J. Zhang, C.-Y. Zhao, and R.-Y. Sun. “Control of tetrahedron satellite formation flying in the geosynchronous orbit using solar radiation pressure”. In: *Astrophysics and Space Science* 361.4 (2016), pp. 1–11. DOI: 10.1007/s10509-016-2732-1 (cit. on p. 181).

[156]X. Huang, Y. Yan, and Y. Zhou. “Optimal spacecraft formation establishment and reconfiguration propelled by the geomagnetic Lorentz force”. In: (2014). DOI: 10.1016/j.asr.2014.08.010 (cit. on p. 181).

[157]O. Montenbruck and E. Gill. “Satellite orbits”. In: Springer-Verlag Berlin Heidelberg, 2000. Chap. 3. ISBN: 978-3-642-58351-3 (cit. on pp. 185, 190, 192, 202).

[158]S. D’Amico and O. Montenbruck. “Proximity Operations of Formation Flying Spacecraft Using an Eccentricity/Inclination Vector Separation”. In: *Journal of Guidance, Control, and Dynamics* 29.3 (2006), pp. 554–563. DOI: 10.2514/1.15114 (cit. on pp. 186, 220).

[159]G. Gaias, J.-S. Ardaens, and O. Montenbruck. “Model of J_2 perturbed satellite relative motion with time-varying differential drag”. In: *Celestial Mechanics and Dynamical Astronomy* 123.4 (2015), pp. 411–433. DOI: 10.1007/s10569-015-9643-2 (cit. on pp. 189, 192).

[160]G. E. Cook. “The effect of aerodynamic lift on satellite orbits”. In: *Planetary and Space Science* 12.11 (1964), pp. 1009–1020. DOI: 10.1016/0032-0633(64)90077-7 (cit. on p. 189).

[161]M. D. Pilinski, B. M. Argrow, and S. E. Palo. “Semiempirical model for satellite energy-accommodation coefficients”. In: *Journal of Spacecraft and Rockets* 47.6 (2010), pp. 951–956. DOI: 10.2514/1.49330 (cit. on p. 189).

[162]V. David and W. D. McClain. *Fundamentals of Astrodynamics and Applications*. Microcosm Press, 2007. Chap. 8. ISBN: 978-1-881883-14-2 (cit. on pp. 190, 191, 193).

[163]L. Mazal, D. Pérez, R. Bevilacqua, and F. Curti. “Spacecraft Rendezvous by Differential Drag Under Uncertainties”. In: *Journal of Guidance, Control, and Dynamics* (2016), pp. 1721–1733. DOI: 10.2514/1.G001785 (cit. on p. 190).

[164]R. H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics*. AIAA education series. American Institute of Aeronautics & Astronautics, 1999. ISBN: 9781600860263. URL: https://books.google.com/books?id=0jH7aVhiGdcC (cit. on pp. 191, 195).

[165]C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. Other Titles in Applied Mathematics. ISBN: 9780898719512. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2000, pp. 279–283 (cit. on p. 193).

[166]M. M. Berry and L. M. Healy. "Implementation of Gauss-Jackson integration for orbit propagation". In: 52 (2004), pp. 331–357. DOI: 10.1.1.618.7461 (cit. on pp. 202, 205).

[167]R. R. Bate, D. D. Mueller, and J. E. White. *Fundamentals of Astrodynamics*. Dover Books on Aeronautical Engineering Series. Dover Publications, 1971. ISBN: 9780486600611. URL: https://books.google.com/books?id=UtJK8cetqGkC (cit. on p. 202).

[168]S. Schweighart and R. Sedwick. "A perturbative analysis of geopotential disturbances for satellite cluster formation flying". In: *Aerospace Conference, 2001, IEEE Proceedings.* Vol. 2. IEEE. 2001, pp. 1001–1019 (cit. on pp. 220, 222).

[169]S. A. Schweighart and R. J. Sedwick. "High-fidelity linearized J2 model for satellite formation flight". In: *Journal of Guidance Control and Dynamics* 25.6 (2002), pp. 1073–1080. DOI: 10.2514/2.4986 (cit. on pp. 220, 222).

[170]M. Pontani and B. A Conway. "Optimal finite-thrust rendezvous trajectories found via particle swarm algorithm". In: *Journal of Spacecraft and Rockets* 50.6 (2013), pp. 1222–1234. DOI: 10.2514/1.A32402 (cit. on p. 220).

[171]S. Wang and C. Zheng. "A hierarchical evolutionary trajectory planner for spacecraft formation reconfiguration". In: *IEEE Transactions on Aerospace and Electronic Systems* 48.1 (2012), pp. 279–289. DOI: 10.1109/TAES.2012.6129635 (cit. on p. 220).

[172]Y. H. Kim and D. B. Spencer. "Optimal spacecraft rendezvous using genetic algorithms". In: *Journal of Spacecraft and Rockets* 39.6 (2002), pp. 859–865. DOI: 10.2514/2.3908 (cit. on pp. 220, 221).

[173]D. Zhang, S. Song, and G. Duan. "Fuel and time optimal transfer of spacecrafts rendezvous using Lambert's theorem and improved genetic algorithm". In: *Systems and Control in Aerospace and Astronautics, 2008. ISSCAA 2008. 2nd International Symposium on*. IEEE. 2008, pp. 1–6. DOI: 10.1109/ISSCAA.2008.4776390 (cit. on p. 220).

[174]R. Storn and K. Price. "Differential evolution. A simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of global optimization* 11.4 (1997), pp. 341–359. DOI: 10.1023/A:1008202821328 (cit. on pp. 220, 226).

[175]M. Iwan, R. Akmeliawati, T. Faisal, and H. M. A. A. Al-Assadi. "Performance comparison of differential evolution and particle swarm optimization in constrained optimization". In: *Procedia Engineering* 41 (2012), pp. 1323–1328. DOI: 10.1016/j.proeng.2012.07.317 (cit. on pp. 221, 242).

[176]J. Vesterstrom and R. Thomsen. “A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems”. In: *Evolutionary Computation, 2004. CEC2004. Congress on*. Vol. 2. IEEE. 2004, pp. 1980–1987. DOI: 10.1109/CEC.2004.1331139 (cit. on pp. 221, 242).

[177]S. P. Lim and H. Haron. “Performance comparison of genetic algorithm, differential evolution and particle swarm optimization towards benchmark functions”. In: *Open Systems (ICOS), 2013 IEEE Conference on*. IEEE. 2013, pp. 41–46. DOI: 10.1109/ICOS.2013.6735045 (cit. on pp. 221, 242).

[178]F. Herrera, M. Lozano, and J. L. Verdegay. “Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis”. In: *Artificial intelligence review* 12.4 (1998), pp. 265–319. DOI: 10.1023/A:1006504901164 (cit. on p. 221).

[179]K. Price, R.M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer Berlin Heidelberg, 2006. ISBN: 9783540313069 (cit. on pp. 221, 226).

[180]D. H. Wolpert and W. G. Macready. “No free lunch theorems for optimization”. In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82. DOI: 10.1109/4235.585893 (cit. on p. 221).

[181]G. Boyarko, O. Yakimenko, and M. Romano. “Optimal rendezvous trajectories of a controlled spacecraft and a tumbling object”. In: *Journal of Guidance Control and Dynamics* 34.4 (2011), pp. 1239–1252. DOI: 10.2514/1.47645 (cit. on p. 221).

[182]Y.-Z. Luo, G.-J. Tang, and H.-Y. Li. “Optimization of multiple-impulse minimum-time rendezvous with impulse constraints using a hybrid genetic algorithm”. In: *Aerospace science and technology* 10.6 (2006), pp. 534–540. DOI: 10.1016/j.ast.2005.12.007 (cit. on p. 221).

[183]A. Ajorlou, K. Moezzi, A. G Aghdam, and S. G. Nersesov. “Two-stage time-optimal formation reconfiguration strategy under acceleration and velocity constraints”. In: *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE. 2010, pp. 7455–7460. DOI: 10.1109/CDC.2010.5717113 (cit. on p. 221).

[184]R. Bevilacqua and M. Romano. “Rendezvous Maneuvers of Multiple Spacecraft Using Differential Drag Under J2 Perturbation”. In: *Journal of Guidance, Control and Dynamics* 31.6 (2008), pp. 1595–1607. DOI: 10.2514/1.36362 (cit. on p. 221).

[185]M. W. Harris and B. Açıkmeşe. “Minimum time rendezvous of multiple spacecraft using differential drag”. In: *Journal of Guidance, Control, and Dynamics* 37.2 (2014), pp. 365–373. DOI: 10.2514/1.61505 (cit. on p. 221).

[186]J. C. Mason and D. C. Handscomb. *Chebyshev Polynomials*. CRC Press, 2002. ISBN: 9781420036114 (cit. on p. 221).

[187]T. J. Rivlin. *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 1990. ISBN: 9780471628965 (cit. on pp. 221, 229).

[188]G. M. Phillips. *Interpolation and Approximation by Polynomials*. CMS Books in Mathematics. Springer New York, 2006. ISBN: 9780387216829 (cit. on pp. 221, 228).

[189]K. E. Atkinson. *An introduction to numerical analysis, 2nd Edition*. Wiley India Pvt. Limited, 2008. ISBN: 9788126518500 (cit. on pp. 221, 228).

[190] C. W. T. Roscoe, J. J. Westphal, J. D. Griesbach, and H. Schaub. "Formation establishment and reconfiguration using differential elements in J2-perturbed orbits". In: *Journal of Guidance, Control, and Dynamics* (2015). DOI: 10.2514/1.G000999 (cit. on p. 223).

[191] C. Louembet. "Design of Algorithms for Satellite Slew Manoeuver by Flatness and Collocation". In: *Proceedings of the 26th American Control Conference* (2007), pp. 3168 –3173. DOI: 10.1109/ACC.2007.4282459 (cit. on p. 224).

[192] S. Das, A. Konar, and U. K. Chakraborty. "Two improved differential evolution schemes for faster global search". In: *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM. 2005, pp. 991–998. DOI: 10.1145/1068009.1068177 (cit. on pp. 226, 227).

[193] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar. "Differential evolution using a neighborhood-based mutation operator". In: *IEEE Transactions on Evolutionary Computation* 13.3 (2009), pp. 526–553. DOI: 10.1109/TEVC.2008.2009457 (cit. on p. 226).

Colophon

This thesis was typeset with LaTeX 2_ε. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.