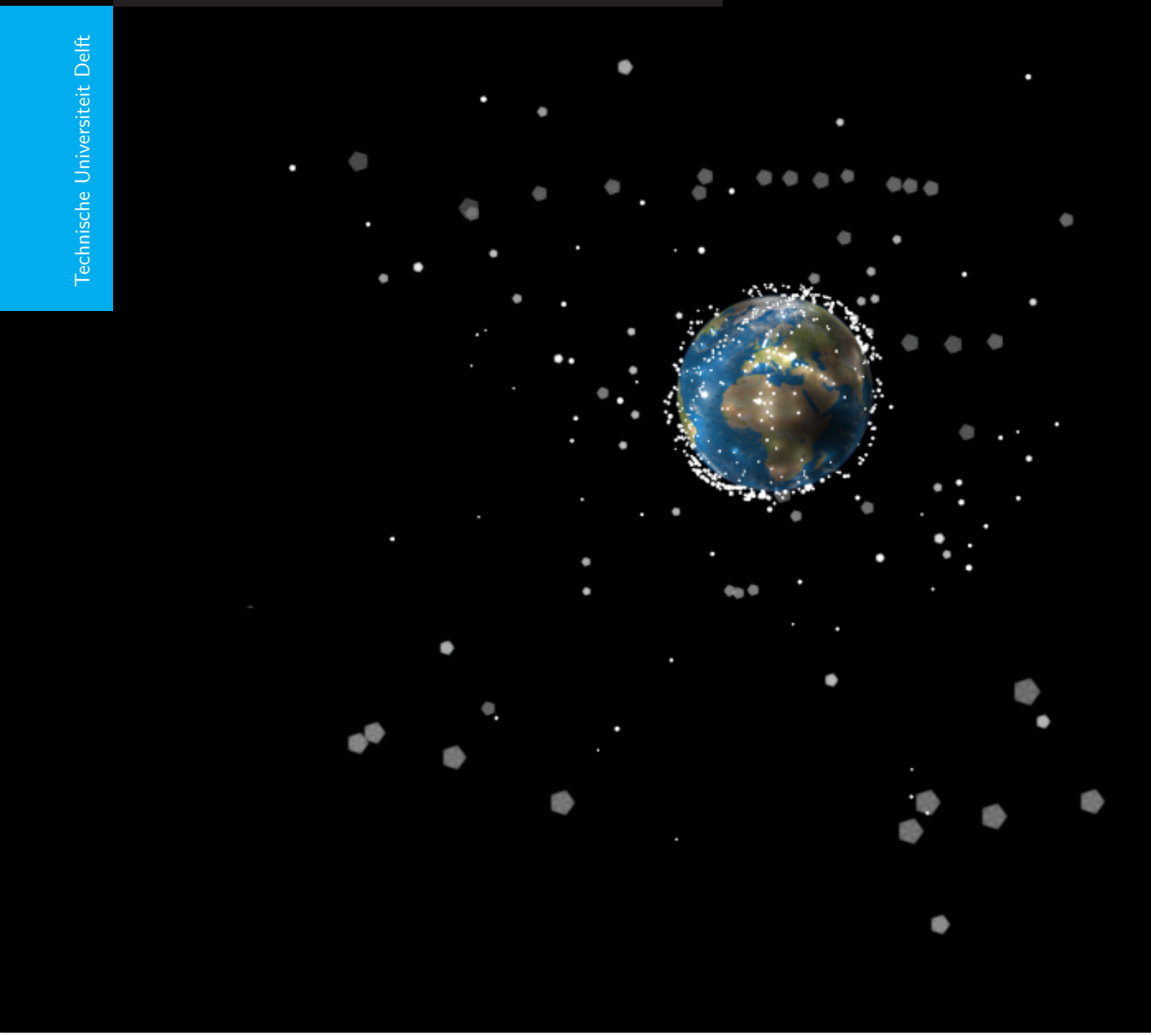
# Variational Multiple Shooting

## Theory and Applications

Cristian Greco

Technische Universiteit Delft



TU Delft Delft University of Technology

Challenge the future

# VARIATIONAL MULTIPLE SHOOTING

## THEORY AND APPLICATIONS

by

**Cristian Greco**

in partial fulfillment of the requirements for the degree of

**Master of Science**

in Aerospace Engineering

at the Delft University of Technology,

to be defended publicly on Friday October 6, 2017 at 14:00.

Committee members:
Dr. E.J.O. Schrama
Ir. R. Noomen (TU Delft Supervisor)
Dr. Ir. S. Hartjes
Prof. Dr. M. Vasile (External Supervisor)


An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

# ACKNOWLEDGEMENTS

This thesis project was carried out at the Aerospace Centre of Excellence of the University of Strathclyde, concluding an ascending *marathon* that led me from my Puglia to Milan, to the Netherlands and finally to Glasgow. As everything in my life, it would be impossible to dissociate my personal experiences from this research adventure, lived with enthusiasm and frustration, with a light mind and through hard work, with hundreds of supporters and alone, with new pals and my solid certainties, always with childish passion. For this reason, before starting the technical discussion I believe we shall dedicate a thought to the most important background, the people who characterized and influenced me, for my desire and their acknowledgement.

Il primo pensiero va alla mia più grande certezza, la mia famiglia, senza la quale sarei solo un navigatore dalle false bussole. A mia mamma Maria, che ringrazia il mio angelo custode ma ancora deve capire che è lei stessa ad esserlo. Grazie per esserci ogni volta e spingermi sempre oltre con forza e dolcezza. A mio papà Mimmo, nelle sue risate ritrovo me stesso, nei suoi silenzi ho capito chi voglio essere. Grazie per la smisurata fiducia. Alle mie sorelle, i cui sguardi significano casa ed amore ovunque noi siamo. A Noemi, alla sua costante presenza ed alla sua voce che mi dona tranquillità. Ad Arianna, con cui ritornerò bambino anche da vecchio e noioso ingegnere. Ai miei nonni, Anna e Gaetano, per avermi donato fieri valori ed un bene generoso, e forse troppi kinder pinguì.

Un ulteriore ringraziamento va alla mia famiglia allargata. A Claudio, mio fratello maggiore acquisito, fatico ad avere un ricordo in cui già non ci fossi. A Matteo, dall'ottimistico potenziale. Agli zii e alle cugine, con i quali son gioiosamente cresciuto e i cui ricordi sono impossibili da cancellare.

A Graziana, il mio elio. Alla nostra tovaglia e al nostro librone, alle nostre paure e ai nostri sogni, sempre crescendo insieme. Grazie per non aver mollato, e per aver dato alla felicità dei contorni ben reali. Il mio orgoglio.

Grazie ai miei amici di una vita, anzi più che amici, Arcangelo, Cosimo, Davide, Gabriele, Lucas, senza i quali sarei un ingegnere migliore ma una persona più vuota. Ai miei compagni di univerisità Andrea, Clarissa, Giuseppe e Stefano, alle innumerevoli ed estenuanti sessioni studio con cibo spazzatura, al divertimento notturno, ai ricordi felici, per un' amicizia che continuerà.

A special thanks to Ron Noomen, my thesis supervisor who supported and checked my work always with enthusiasm and kindness. A further thanks to all the people in the Aerospace Centre of Excellence, with a special regard for my external supervisor Max, Annalisa and Romain, for the wise advices and for making me feel part of the group from the very first day.

Ed anche a me stesso, che lotto sempre per restare sveglio.

*Cristian Greco*
*Delft, October 2017*

# CONTENTS

# NOMENCLATURE

| Symbol | Description |
|---|---|
| $ACE$ | Aerospace Centre of Excellence |
| $AU$ | Astronomical Unit |
| $BFGS$ | Broyden–Fletcher–Goldfarb–Shanno algorithm |
| $BVP$ | Boundary Value Problem |
| $CNEOS$ | Center for Near-Earth Object Studies |
| $ECEF$ | Earth-Centered Earth-Fixed frame |
| $ECI$ | Earth-Centered Inertial frame |
| $ESA$ | European Space Agency |
| $GTOC9$ | 9th Global Trajectory Optimization Competition |
| $HM$ | Heuristic Method |
| $IP$ | Interior point algorithm |
| $IVP$ | Initial Value Problem |
| $JD$ | Julian Date |
| $JAXA$ | Japan Aerospace Exploration Agency |
| $JPL$ | Jet Propulsion Laboratory |
| $KKT$ | Karush Kuhn Tucker |
| $LEO$ | Low Earth Orbit |
| $LHS$ | Latin Hypercube Sampling |
| $LTTT$ | Low-Thrust Trajectory Tools Team |
| $MJD$ | Modified Julian Date |
| $NASA$ | National Aeronautics and Space Administration |
| $NEA$ | Near-Earth Asteroid |
| $NEO$ | Near-Earth Object |
| $NLP$ | Non-Linear Programming |
| $ODE$ | Ordinary Differential Equation |
| $OCP$ | Optimal Control Problem |
| $RK$ | Runge-Kutta |
| $SMART$ | Strathclyde Mechanical and Aerospace Research Toolbox for Astrodynamics |
| $SQP$ | Sequential Quadratic Programming |
| $TOF$ | Time Of Flight |
| $TPBVP$ | Two-Point Boundary Value Problem |
| $WORHP$ | Worhp Optimises Really Huge Problems |

Table 1: List of Acronyms.

| Symbol | Description | Unit |
|---|---|---|
| $a$ | Semi-major axis | m |
| $\mathbf{a}$ | Equality constraint | - |
| $\mathbf{a}_d$ | Perturbative acceleration | m/s$^2$ |
| $\mathbf{b}$ | Inequality constraint | - |
| $\mathbf{c}$ | Constraint vector | - |
| $DF$ | Objective function gradient vector | - |
| $DG$ | Jacobian of the constraints matrix | - |
| $e$ | Eccentricity | - |
| $\mathbf{e}_r$, $\mathbf{e}_t$, $\mathbf{e}_n$, $\mathbf{e}_h$ | Unit vectors in radial, transversal, normal and out-of-plane directions | - |
| $\mathbf{f}(t,\mathbf{x},\mathbf{u})$ | Dynamical equations | - |
| $\mathbf{f}_P$ | Generic perturbative specific force vector | m/s$^2$ |
| $\mathbf{f}_T$ | Specific thrust vector | m/s$^2$ |
| $g^f$ | Fixed-time constraint | - |
| $g^f_L$, $g^f_U$ | Fixed-time constraint lower and upper bounds | - |
| $g^P$ | Path constraint | - |
| $g^P_L$, $g^P_U$ | Path constraint lower and upper bounds | - |
| $\tilde{g}$ | Scaled constraint | - |
| $\mathbf{g}$ | Constraint vector / Objective gradient | - |
| $\widetilde{G}$ | Scaled Jacobian of the constrains | - |
| $\mathbf{G}$ | Jacobian of the constrains | - |
| $h_e$, $h_m$ | Altitude over Earth and Moon surface | m |
| $H$ | Hamiltonian | - |
| $\mathbf{H}$ | Hessian matrix | - |
| $HM$, $\mathbf{H}_{\mathscr{L}}$ | Hessian matrix of the Lagrangian | - |
| $i$ | Inclination | rad |
| $I_{sp}$ | Specific impulse | s |
| $J$, $\widetilde{J}$ | Cost functions | - |
| $\mathbf{J_c}$ | Jacobian of the constraints matrix | - |
| $J_{m,n}$ | Harmonic expansion coefficient | - |
| $L$ | Integral term in objective function | - |
| $\mathscr{L}$ | Lagrangian | - |
| $m$ | Mass / Number of discretization points | kg / - |
| $m_c$ | Number of constraints g | - |
| $m_{de}$, $m_{dry}$, $m_p$ | De-orbit package, dry and propellant mass | kg |
| $n_c$ | Number of control components | - |
| $n_p$ | Number of control parameters per control component | - |
| $n_s$ | Number of state variables | - |
| $n_y$ | Number of free parameters | - |
| $n_{\Delta V}$ | Number of $\Delta V$ per transfer | - |
| $\mathscr{O}$ | Landau's O | - |
| $\tilde{p}_j$ | Generic scaled free parameter | - |
| $p_j$ | Generic free parameter | - |
| $\mathbf{p}$ | Search direction | - |
| $P_{n,m}$ | Legendre polynomial | - |
| $r$ | Position vector magnitude | m |
| $\mathbf{r}$ | Position vector | m |

Table 2: List of Roman symbols - 1.

| Symbol | Description | Unit |
|---|---|---|
| $\dot{\mathbf{r}}$ | Velocity vector | m/s |
| $\ddot{\mathbf{r}}$ | Acceleration vector | m/s$^2$ |
| $r_{eq}$ | Central body mean equatorial radius | m |
| $R$ | Radius shaping in spherical shaping | m |
| $\mathbb{R}$ | Set of real numbers | - |
| $s_j$ | Scaling factor | - |
| $S^k_{p_j}$ | Sensitivity of $x_k$ w.r.t. $p_j$ | - |
| $\mathbf{S}_H$ | Second-order sensitivity matrix | - |
| $\mathbf{S}_u$ | First-order sensitivity matrix w.r.t. control parameters | - |
| $\mathbf{S}_x$ | First-order sensitivity matrix w.r.t. state parameters | - |
| $t$ | Time | s |
| $t_0$, $t_f$ | Initial and final time | s |
| $t_q$ | Gauss-Legendre node | - |
| $u_k$ | Control variable | - |
| $\mathbf{u}$ | Control vector | m/s$^2$ |
| $\tilde{\mathbf{u}}$ | Scaled control vector | m/s$^2$ |
| $w_q$ | Gauss-Legendre node weight | - |
| $\mathbf{W}$ | Diagonal scaling matrices | - |
| $x_k$ | State variable | - |
| $\mathbf{x}$ | State vector | - |
| $\mathbf{x}_g$ | Guess of state vector | - |
| $\mathbf{x}_p$ | Propagated state vector | - |
| $\tilde{\mathbf{x}}$ | Scaled state vector | - |
| $\mathbf{y}$ | Free parameters vector | - |

Table 3: List of Roman symbols - 2.

| Symbol | Description | Unit |
|---|---|---|
| $\alpha$ | Scaling factor for GTOC9 objective function | [M€/kg$^2$] |
| $\beta$ | Objective function for interior point algorithms | - |
| $\delta$ | Generic perturbation | - |
| $\delta_{i,j}$ | Kronecker delta of variables i and j | - |
| $\Delta$ | Variation | - |
| $\Delta J$ | Objective function gradient vector | - |
| $\Delta v$, $\Delta V$ | Indicator of propellant consumption | m/s |
| $\theta$ | True anomaly | rad |
| $\lambda$, $\boldsymbol{\lambda}$, $\mu$, $\boldsymbol{\mu}$ | Lagrangian multipliers | - |
| $\mu$ | Gravitational parameter | m$^3$/s$^2$ |
| $\nabla$ | Nabla | - |
| $\tau$ | Scaled time | s |
| $\tau_0$ | Time of periapsis passage | s |
| $\varphi$,$\theta$ | Spherical angular coordinates | rad |
| $\phi$, $\Phi$ | End-cost term | - |
| $\Phi$ | Angular shaping in spherical shaping | rad |
| $\omega$ | Argument of periapsis | rad |
| $\Omega$ | Right ascension of the ascending node | rad |

Table 4: List of Greek symbols.

# 1

# INTRODUCTION

Low-thrust propulsion has nowadays found wide application as primary thrust in the near-Earth environment for final orbit insertion, orbital transfers, station-keeping and satellite disposal. Electric propulsion entails considerable savings in the propellant mass thanks to the very high specific impulse which they are able to generate. However, potential profits are always accompanied by corresponding complications. The mathematical problem of an orbital trajectory becomes very complex when low-thrust propulsion is considered. Only in the last decades augmented computational capabilities and progress in optimization techniques, as well as the development of more efficient engines, made interplanetary trajectories affordable and optimally designable. Remarkable cases are the pioneering NASA's Deep Space 1, JAXA's Hayabusa and ESA's SMART-1 completed missions, coupled with NASA's Dawn and ESA's BepiColombo currently ongoing and future missions [1]. However, this still represents a quite immature sector, when compared to other space disciplines, with a prosperous research field aiming to extrapolate all of its potential.

In 2002, NASA founded a research team, namely the Low-Thrust Trajectory Tools Team (LTTT) [1], to create a toolbox of optimization software with variable levels of final accuracy and computational load. However, the distribution of these tools is limited to the different NASA centers and only the results of their application to a number of test scenarios were published. Several institutions and space agencies have also developed their own tools, but again their availability typically stops at the institution's internal boundaries. In addition, various tools with a low-medium fidelity level for low-thrust optimization have been created by university researchers and several of them are open source.

The research goal of the present thesis work is the development of a novel multiple-shooting tool for space trajectory optimization and the assessment of its performance and accuracy in a variety of practical and challenging test cases, such as a CubeSat-asteroid rendezvous feasibility study, and *The Kessler Run*, the 9th Global Trajectory Optimization Competition (GTOC9) problem. The main novelty of this project consists in the variational approach for derivative computation. The theory behind this technique, only outlined in a few references [2] [3], has not been described in detail, as well as the performance of this method has not been widely investigated. Hence, this thesis aims to help filling this gap in literature by developing the theory of the variational equations for optimization problems, and by assessing the efficiency of this approach. This project will be carried out in the Aerospace Centre of Excellence (ACE), space research laboratory of the University of Strathclyde.

The report is structured in seven chapters, including the present introduction, following a sequential flow of required concepts. Chapter 2 recalls fundamental concepts of celestial mechanics

needed for motivating subsequent implementation choices, given the tool's specific purpose, and for analyzing the test cases' outcomes. Chapter 3 discusses optimal control theory and practical methods to solve it, topics which represent the key background knowledge on top of which the novel algorithm shall be developed. By comparing different transcription schemes, the selection of a multiple-shooting algorithm will be justified. It will be shown why the classical derivative computation technique is the bottleneck of this method, and the variational approach will be identified as a potential alternative to improve this computational burden. In Chapter 4, this topic will be examined in depth from a theoretical standpoint by deriving the equations of the variational dynamics. This central part of the report represents the key theoretical fulcrum of the novel approach developed in this thesis. On the other hand, Chapter 5 represents the numerical counterpart of the previous one, being concerned about the practical details of the tool implementation and its verification and validation against a number of elementary test cases. In particular, the validation phase will represent the first assessment of the variational approach performance in comparison with the finite-difference routine. Once tested, the variational multiple shooting will be used to solve two different complex test cases, as reported in Chapter 6. First, a single-phase low-thrust CubeSat-asteroid rendezvous will be optimized and the results analyzed to evaluate the feasibility of such a mission. Then, the tool will be employed as local optimizer in an optimization cascade to solve the GTOC9 challenge with the Strathclyde++ team. For both applications, the performance of the variational approach will be further examined. Finally, Chapter 7 will summarize and unify the research outcomes discussed in the previous chapters, and introduce ideas for future research developments.

# 2

# ASTRODYNAMICS

The astrodynamics background theory developed in this chapter will represent the ground level of knowledge on which the optimization tool, aimed for specific space applications, is built. The implementation choices motivated in Chapter 5, and the test cases analyzed in Chapter 6, will often take advantage of orbital mechanics concepts introduced in these sections, where both fundamental and well-known topics, such as reference frames and coordinate systems, and specific tools, such as low-fidelity inverse methods, will be shortly outlined.

In particular, both interplanetary trajectories (see Section 6.1) and Earth transfer orbits (see Section 6.2) will be studied as test cases, which employ different reference frames, i.e. inertial heliocentric and Earth-centered respectively, as introduced in Section 2.1. Some basic orbital mechanics and the principal perturbations in near-Earth environment will be briefly reported in Section 2.2, as they are directly needed for the applications discussed in Chapter 6 as well. This discussion will prove useful to understand why some perturbations shall not be neglected in the computation of specific trajectories when a high-accuracy outcome is sought. Lastly, shaping methods will be introduced in Section 2.3 as methods to generate first-guess solutions needed for any numerical optimization tool, as pointed out in Section 3.2. A particular focus will be given to the spherical shaping as it will generate the first guess for the CubeSat rendezvous test case.

## 2.1. REFERENCE FRAMES AND COORDINATE SYSTEMS

Reference frames and coordinate systems are milestone topics in any space-related study. In physics, their definition is required to describe position and velocity of any general body, such as planets, asteroids or spacecraft in the space scenario. Although completely equivalent by means of proper transformations, the form of the equations governing their motion are dramatically dependent on the selected frame and coordinate system, and selecting the right one is crucial to have a convenient form for the problem in study. [4] dedicates an extensive chapter to reference frames and coordinate systems, which will be used as main reference for the following section.

### 2.1.1. REFERENCE FRAMES

The fundamental and most precise celestial reference frame nowadays is the *International Celestial Reference System*, which has been developed by the space community taking into account also relativity effects. The origin of this system is at the barycenter of the Solar System and the axes point towards very distant celestial objects, such as quasars or nuclei of galaxies, with negligible relative

angular motion to make it quasi-inertial (in practice it is impossible to select a perfectly inertial frame). However, when simulating an interplanetary trajectory as a two-body motion around the Sun, this is not the most convenient one because the origin does not precisely coincide with the Sun's center of mass and the motion of our star shall be taken into account.

In practical applications, a *non-rotating heliocentric frame* is used for interplanetary trajectories' studies. The reference XY-plane of Sun-centered frames is the ecliptic plane, i.e. the plane coplanar to Earth's orbit around the Sun. Then, the +X-axis is chosen to point to the First Point of Aries, i.e. the vernal equinox, at a chosen date. The most common choice is the position of the vernal equinox at 12:00 of 01/01/2000 (J2000). The +Z-axis is chosen to form an acute angle with the direction of the celestial Earth north pole. The +Y-axis then completes the right-hand frame within the reference plane.

Earth-orbiting spacecraft are usually described in the *non-rotating geocentric equatorial reference frame*, or *Earth-centered inertial frame* (ECI). The reference frame is centered in Earth's center of mass and, as implied by the name, the XY-plane is coplanar with the equator. The +X-axis is chosen again to point towards the vernal equinox (usually J2000), while the +Z-axis points towards Earth's celestial north pole. The +Y-axis lies in the equatorial plane and it forms a right-hand frame with the already defined axes.

An often-employed reference frame for Earth's satellites is the *geocentric rotating reference frame*, also known as *Earth-centered Earth-fixed* (ECEF). The origin is still the Earth's barycenter and the +Z-axis again points towards the Earth's celestial north pole, as well as the definition of the XY-plane remains unaltered. Differently, the +X-axis always crosses the Greenwich meridian and therefore it is rotating, while the +Y-axis completes the right-handed frame.

### 2.1.2. COORDINATE SYSTEMS

Two main types of coordinate systems are used in celestial mechanics. The *Cartesian* representation uses the projections $X, Y, Z$ of the spacecraft position vector on the reference frame axes and their time-derivatives. A drawback of this system is that all the coordinates are fast variables, i.e. they change significantly and rapidly during orbital revolution. On the contrary, the *orbital elements* representation takes advantage of the integrals of motion of the two-body problem to describe the position and velocity of a spacecraft by means of unperturbed quantities plus only one fast variable.

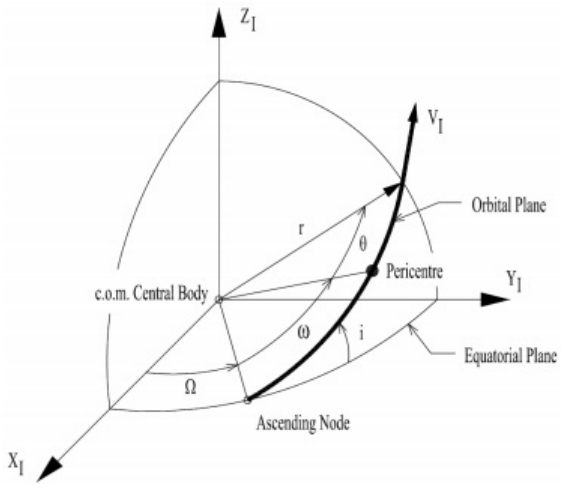



Figure 2.1: Illustration of an orbital plane with focus on inclination, argument of periapsis and longitude of the ascending node [5].

The classical set of orbital elements is the *Keplerian* one, which is composed by $a$, $e$, $i$, $\omega$, $\Omega$ and $\theta$, respectively the semi-major axis, eccentricity, inclination angle, argument of pericenter, right ascension of the ascending node and true anomaly of the satellite within the orbit at a reference epoch $\tau_0$.

## 2.2. DYNAMICAL MODELS

This section will serve as a brief recap of the dynamical equations governing the orbital motion and which will be used for the tool applications in Chapter 6 .

### 2.2.1. PERTURBED TWO-BODY MOTION

[6] defines perturbations as the *"forces acting on an object other than those forces that cause it to move on some reference orbit"*. The reference trajectory typically considered in orbital mechanics is the two-body motion of a spacecraft with negligible mass around a central body, modeled as a uniform sphere. It results from the following differential equation:

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{\mu}{r^3}\mathbf{r} \tag{2.1}$$

where the vector $\mathbf{r}$ represents the relative position between the two bodies, $r$ its modulus and $\mu$ is the central body gravitational parameter [4]. This system can be analytically integrated as a function of an angular quantity, usually the true anomaly $\theta$. The resulting trajectory is the equation of a conic as follows:

$$r = \frac{a(1-e^2)}{1+e\cos\theta} \tag{2.2}$$

Every perturbation, regardless of its magnitude, can be plugged in the right-hand side of Equation (2.1), modifying it as follows:

$$\frac{d^2\mathbf{r}}{dt^2} = -\frac{\mu}{r^3}\mathbf{r} + \mathbf{f}_P \tag{2.3}$$

The ratio between the perturbing acceleration and the local central gravitational acceleration values shall be at least of order $\mathcal{O}(10^{-2})$ to properly apply the perturbative approaches. A similar order of magnitude shall apply to consider the trajectory generated by a shaping method (see Section 2.3), based only on the two-body pull, as a suitable initial guess for trajectory optimization with more complex dynamics. A comparison of the magnitude of different perturbation sources in the near-Earth environment is plotted in Figure 2.2.

For the present research project, the perturbations considered in the force model depend on the test case specifics. Hence, the focus of this section will only be on the nature and representation of perturbation forces which will later be employed in the tool's test cases, and not on their effect on the orbital elements, which strongly depends on the chosen reference trajectory characteristics.

### 2.2.2. GRAVITATIONAL OBLATENESS

The differential Equation (2.1) has been obtained by the approximation of the central body as a uniform sphere. However, celestial bodies have neither a radially symmetric mass distribution nor a perfect spherical shape. By solving Laplace's equation in spherical coordinates imposed for the

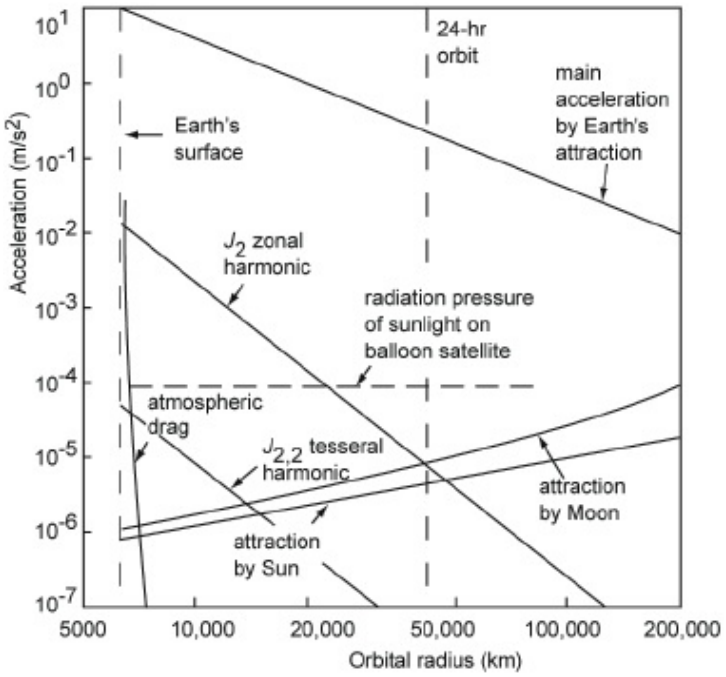



Figure 2.2: Magnitude of perturbing forces as a function of orbital radius [4].

gravitational potential of continuous mass distribution, as [7] explains, we can express the perturbative gravitational potential of a celestial body in an inertial centered reference frame as a series of *Legendre polynomials* $P_{n,m}$:

$$\tilde{R} = \frac{\mu}{r}\left[\sum_{n=2}^{\infty} J_n \left(\frac{r_{eq}}{r}\right)^n P_n(\sin\phi) + \sum_{n=2}^{\infty}\sum_{m=1}^{n} J_{n,m}\left(\frac{r_{eq}}{r}\right)^n P_{n,m}(\sin\phi)\{\cos m(\Lambda - \Lambda_{n,m})\}\right] \tag{2.4}$$

where $r_{eq}$ is the mean radius of the central body, $\phi$ and $\Lambda$ the angular spherical coordinates, $J_{n,m}$ and $\Lambda_{n,m}$ are model parameters. The main effect is given by the first perturbing term $J_2$, the disturbance considered in the test case in Section 6.2, while the other coefficients have much smaller orders of magnitude. By taking the spatial gradient of the potential in Equation (2.4), the resulting acceleration is modeled by the following dynamical equations [4]:

$$\begin{aligned} f_{P_x} &= -\frac{3}{2}\mu J_2 \frac{r_{eq}^2}{r^5} x\left(1 - 5\frac{z^2}{r^2}\right) \\ f_{P_y} &= -\frac{3}{2}\mu J_2 \frac{r_{eq}^2}{r^5} y\left(1 - 5\frac{z^2}{r^2}\right) \\ f_{P_z} &= -\frac{3}{2}\mu J_2 \frac{r_{eq}^2}{r^5} z\left(3 - 5\frac{z^2}{r^2}\right) \end{aligned} \tag{2.5}$$

### 2.2.3. THIRD-BODY PERTURBATION

The third-body disturbance is caused by the gravitational interaction of a third celestial body with the spacecraft. Perturbation's resulting acceleration is derived in rectangular coordinates in an inertial central body reference frame as [4]:

$$\mathbf{f}_p = \mu_j \left( \frac{\mathbf{r}_D - \mathbf{r}_i}{r_{iD}^3} - \frac{\mathbf{r}_D}{r_D^3} \right) \tag{2.6}$$

where $\mu_D$ is the standard gravitational parameter of the third body, $\mathbf{r}_D$ is the vector distance between the third and the central body, $\mathbf{r}_i$ the vector distance from the central body to the spacecraft, and $\mathbf{r}_{iD}$ their difference, i.e. the position vector from the spacecraft to the perturbing body. This acceleration model will be employed in Section 6.1, when the effect of the Earth and the Moon on an interplanetary trajectory will be considered.

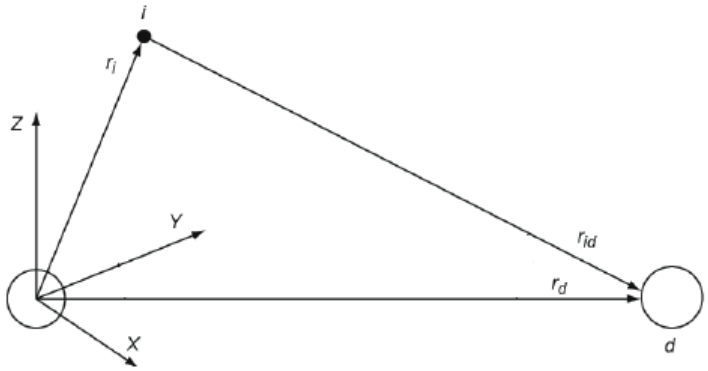



Figure 2.3: Illustration of third-body perturbation [4].

## 2.3. INVERSE SHAPING METHODS

The design of low-thrust trajectories has been historically treated as an optimization problem in literature (see Section 3.1). However, all methods developed nowadays are computationally expensive due to the enormous number of degrees of freedom, and often require an initial estimate of the optimal trajectory. The multiple-shooting tool developed for the current research does no exception. To compute first-guess solutions, the so-called shape-based methods model analytically the geometry of the trajectory as well as its time evolution. The control profile is then obtained by *inverting* the equations of motion, i.e. requiring the following equality to be satisfied all along the trajectory:

$$\mathbf{f}_T = \ddot{\mathbf{r}} + \frac{\mu}{r^3}\mathbf{r} \tag{2.7}$$

where $\mathbf{f}_T$ is the thrust acceleration. The environmental perturbations have been left out from this formulation in any of the developed shaping methods as the goal is to develop a first-order solution. The parameters within the assumed shape shall be chosen such that the boundary conditions are satisfied. If some degrees of freedom are left, an optimization of the remaining parameters could be carried forward to minimize a performance index and/or impose constraints on the maximum peak thrust and the time of flight (TOF) feasibility [8].

[9] developed the very first shaping method, named exposin as it was employing exponential sinusoidal functions to parameterize the radius. However, as major drawbacks, this method does not satisfy the boundary conditions on velocity, often leading to unfeasible solutions, and the described motion is only planar. Later developments of new shaping frameworks managed to overcome these restrictions. In the current thesis, only the spherical shaping method, discussed in the reminder of the section, have been employed for its advantageous characteristics as well as its availability within the Aerospace Centre of Excellence, the research center where the project has been carried out.

### 2.3.1. SPHERICAL SHAPING

[10] reformulated the state of the spacecraft as function of the spherical coordinates $(r,\theta,\phi)$ to describe its 3D motion.

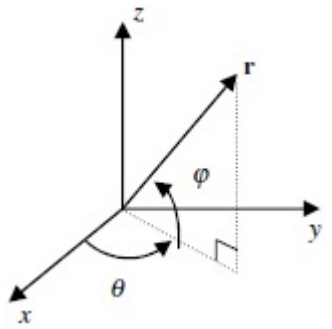



Figure 2.4: Spherical coordinate system [10].

To describe the trajectory, time is substituted by the angular variable $\theta$ as independent variable. The state then becomes $[r,t,\varphi,r',t',\varphi']$, where the prime $'$ points out the derivative with respect to $\theta$. Inverting the equations of motion, the thrust control in tangential-normal-out-of-plane $(\boldsymbol{e_t},\boldsymbol{e_n},\boldsymbol{e_h})$ frame assumes the form [10]:

$$f_T = \begin{cases} f_{T_t} = \frac{\mu}{r^2}\mathbf{e_r}\cdot\mathbf{e_t} + \ddot{\theta}\tilde{\mathbf{v}}\cdot\mathbf{e_t} + \dot{\theta}^2\tilde{\mathbf{a}}\cdot\mathbf{e_t} \\ f_{T_n} = \frac{\mu}{r^2}\mathbf{e_r}\cdot\mathbf{e_n} + \dot{\theta}^2\tilde{\mathbf{a}}\cdot\mathbf{e_n} \\ f_{T_h} = \dot{\theta}^2\tilde{\mathbf{a}}\cdot\mathbf{e_h} \end{cases} \tag{2.8}$$

where $\tilde{\mathbf{v}} = d\mathbf{r}/d\theta$ and $\tilde{\mathbf{a}} = d^2\mathbf{r}/d\theta^2$. The state vector is parameterized with a functional form as $[r = R(\theta), t = T(\theta), \varphi = \Phi(\theta)]$ such that the normal component of the control vector vanishes, i.e. Equation (2.8-2). First, a shaping function for the time $\theta$-derivative is given:

$$T' = \sqrt{\frac{DR^2}{\mu}} \tag{2.9}$$

where $D = -r'' + 2\frac{r'^2}{r} + r'\varphi'\frac{\varphi''-\sin\varphi\cos\varphi}{\varphi'^2+\cos^2\varphi} + r(\varphi'^2 + cos^2\varphi)$ is a geometrical parameter. The choice of shaping the derivative of time $T'$ arises from the impossibility to obtain a near-optimum control given an a priori shaping function $T$. In the latter case indeed, the control usually exceeds the low-thrust limits to fulfill the prescribed path, while $T'$ in Equation (2.9) has proven to ensure a feasible thrust profile [10].

The shaping functions $R$ and $\Phi$ are selected such that the parameters can satisfy the boundary conditions through analytic non-iterative computations. There are four boundary conditions on $R$ and $R'$, four on $\Phi$ and $\Phi'$ and two on $T'$. Given the expression of $T'$, the last two boundary conditions can be translated as conditions on $R''$ and $\Phi''$. Therefore, the shaping functions for the radius and the elevation angles shall have at least ten free parameters.

$$\begin{aligned} R &= \frac{1}{a_0 + a_1\theta + a_2\theta^2 + (a_3 + a_4\theta)\cos\theta + (a_5 + a_6\theta)\sin\theta} \\ \Phi &= (b_0 + b_1\theta)\cos\theta + (b_2 + b_3\theta)\sin\theta \end{aligned} \tag{2.10}$$

The eleventh parameter is used to satisfy an extra constraint on the time of flight.

Novak developed a general formulation of the shaping methods in his PhD thesis, better formalizing the mathematical basis of the inverse methods, which can be used as mathematical reference. The parameterization choice, in Equation (2.10), ensures that the minimum-thrust arc is the Keplerian one, a suitable choice for reproducing coasting arcs. According to several test cases described in [10], this approach generates near-optimal solutions in terms of $\Delta V$ for interplanetary trajectories, and therefore it is a convenient method for first-guess generation during the asteroid rendezvous application which will be analyzed in Section 6.1.

# 3

# OPTIMIZATION

The goal of the thesis project concerns the development of a numerical tool for space trajectory optimization employing a variational approach to enhance the computational performance of derivative computation. However, before starting to develop the variational theory and implement the tool, we need to answer a few questions first. How shall a space trajectory optimization problem be formulated? What is the theory behind continuous optimization? Which are the best practical methods to solve space problems of interest for this thesis? How do numerical algorithms work? The present chapter aims to introduce the key background knowledge needed for the current research development, as well as the reasoning behind the choice of particular research paths.

The generic continuous control problem, with an objective function to minimize and constraints to respect, will be framed under the general formulation of *optimal control problem* (OCP) theory as described in Section 3.1. Here, the differences between direct and indirect methods to solve OCP will be analyzed and a trade-off between them performed. Since these techniques lead to a closed-form solution only for elementary cases, practical methods to find an approximated solution will be treated in Section 3.2. Any of these converts the infinite-dimension optimization problem to a finite-dimension one through a transcription method which dictates the parameters to optimize. It will be shown that once the set of free parameters is selected, the transcribed optimal control method is posed as a nonlinear programming (NLP) problem, of which different solution approaches will be addressed in Section 3.3. Global optimization methods fall outside the *optimal control theory* and are not strictly related to the main objectives of this thesis. However, they usually find applicability as a previous step or on an outer optimization-loop level with respect to the local refinement algorithms discussed above. For example, they can be employed to optimize static parameters such as the departure date and time of flight of a rendezvous trajectory (see Section 6.1). Therefore, Section 3.4 will give a brief mention to some of these techniques and discuss their applicability.

## 3.1. OPTIMAL CONTROL THEORY

The general statement of an *optimal control problem* (OCP) requires the definition of [11]:

- The mathematical model of the dynamic system to control.
  Usually it is described by a system of ordinary differential equations (ODEs) in the form $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t))$. The independent variable has been indicated by $t$, usually appointed as time, but there is no restriction on its choice. The variables $x_i$ in the vector of $\mathbf{x}$ are usually called *state variables*, while $u_j$ in the vector $\mathbf{u}$ are the *control variables*.

- The performance index $J$ to be minimized (or equivalently maximized).
  The performance index in the general form is written as:

$$J = \phi[t_f, \mathbf{x}(t_f)] + \int_{t_0}^{t_f} L[t, \mathbf{x}(t), \mathbf{u}(t)]\, dt \tag{3.1}$$

  The *optimal control problem* is in the Bolza form if both the end-cost and the integral terms are present. If the end-cost term $\phi$ is zero, it is known as a Lagrange problem. On the contrary, if the integral term $L$ is zero, the problem is referred as a Mayer one. Mathematically these formulations are equivalent and convertible into each other. For example, a Lagrange problem can be restated as a Mayer one simply adding one state variable of the form $\dot{x}_{n+1} = L[t, \mathbf{x}(t), \mathbf{u}(t)]$, leading to $J = x_{n+1}(t_f)$. However, [3] states that, even if they are mathematically equivalent, they are not numerically corresponding. The Lagrange form shall be preferred as the Mayer form leads to an increased number of state variables, which are then discretized in numerical methods, leading to a higher size of the NLP subproblem and a more time-consuming algorithm. The most common choices for the performance index in low-thrust problems are the total $\Delta v$ of the trajectory (or equivalently the final mass) for electric engines, and time of flight for solar sails, as the thrust is generated by an external source which is potentially infinite in the latter.

- Specification of constraints.
  They are divided into two different classes, i.e. *fixed-event* or *path* constraints. The first type is described as an algebraic function of the state and control $g_L^f \le g^f[(\bar{t}_j), \mathbf{y}(\bar{t}_j), \mathbf{u}(\bar{t}_j)] \le g_U^f$ at a fixed time $\bar{t}_j$. The initial and final boundary conditions fall into this form for $g_L^f = g_U^f$. A *path constraint* is formulated as an algebraic function of the state and control variables $g_L^p \le g^p[(t), \mathbf{x}(t), \mathbf{u}(t)] \le g_U^p$ over a trajectory's phase. Bounds on the control magnitude fall into this category as $\mathbf{u}_L \le \mathbf{u}(t) \le \mathbf{u}_U$. This general notation [3] deals with both equality and inequality constraints, depending on the lower and upper boundary values.

Once the aforementioned statements have been formulated, the *optimal control problem* aims to find the control profile $\mathbf{u}^*(t)$, in the space of all admissible controls $U$, which minimizes the performance criterion $J$ while respecting the differential model $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t))$ and the specified physical constraints. Briefly stated:

$$\begin{aligned} &min \;\; J = \phi[t_f, \mathbf{x}(t_f)] + \int_{t_0}^{t_f} L[t, \mathbf{x}(t), \mathbf{u}(t)]\, dt, \quad u \in U \\ &subject\ to: \dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)) \\ &\qquad \mathbf{g}_L^p \le \mathbf{g}^p[(t), \mathbf{x}(t), \mathbf{u}(t)] \le \mathbf{g}_U^p \\ &\qquad \mathbf{g}_L^f \le \mathbf{g}^f[(\bar{t}_j), \mathbf{x}(\bar{t}_j), \mathbf{u}(\bar{t}_j)] \le \mathbf{g}_U^f \end{aligned} \tag{3.2}$$

where the Bolza formulation is used to obtain the necessary conditions in the most general case.

### 3.1.1. INDIRECT METHODS

Indirect methods are based on Pontryagin's maximum principle, adapting the sign convention for minimization problem. This principle's derivation employs *calculus of variations* techniques, of which comprehensive references are [12] and [13]. The goal is to convert the *optimal control problem* as defined in the chapter's introduction into a *two-point boundary value problem* through the statement of the necessary conditions that a profile shall satisfy to be an optimal solution.

The process starts with the definition of an *augmented performance index* $\bar{J}$, in a fashion similar to equality-constrained static optimization problems, where Lagrange's multipliers $\lambda_j$ multiplying the dynamical constraints are summed to the objective function to form the *augmented performance index*:

$$\bar{J} = \Phi + \int_{t_0}^{t_f} \Big[ L[t, \mathbf{x}(t), \mathbf{u}(t)] + \boldsymbol{\lambda}^T(t) \{ \mathbf{f}[t, \mathbf{x}(t), \mathbf{u}(t)] - \dot{\mathbf{x}} \} \Big] dt \tag{3.3}$$

According to *calculus of variation*, the necessary conditions for a stationary extremum is that the first-order variation $\delta \bar{J}$ shall nullify at any instant of time for any constraint-allowed variation $\delta \mathbf{u}(t)$. The problem *Hamiltonian* is defined as:

$$H = L[t, \mathbf{x}(t), \mathbf{u}(t)] + \boldsymbol{\lambda}^T(t) \mathbf{f}[t, \mathbf{x}(t), \mathbf{u}(t)] \tag{3.4}$$

When path constraints are present, the Hamiltonian shall be augmented with the constraints' violation weighted by associated dual variables. After mathematical manipulation (see [11] for a detailed derivation), the necessary conditions for a control profile $\mathbf{u}^*(t)$ to be a stationary function of the performance index are represented by the following *Euler-Lagrange equations*:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)) \\ \dot{\boldsymbol{\lambda}} &= -\left[ \frac{\partial H}{\partial \mathbf{x}} \right]^T \\ \mathbf{0} &= \left[ \frac{\partial H}{\partial \mathbf{u}} \right]^T \end{aligned} \tag{3.5}$$

where the relations in Equation (3.5)-2 are labeled as *adjoint equations* and Equations (3.5)-3 as *control equations*. These differential equations, which a control profile has to necessarily satisfy to be a stationary solution, are coupled with a set of *transversality conditions*:

$$\begin{aligned} t_0 \;\; given \quad &\vee \quad H(t_0) = 0 \\ t_f \;\; given \quad &\vee \quad H(t_f) = -\frac{\partial \Phi}{\partial t}\Big|_{t_f} \\ \mathbf{x}(t_0) \;\; given \quad &\vee \quad \boldsymbol{\lambda}(t_0) = 0 \\ \mathbf{x}(t_f) \;\; given \quad &\vee \quad \boldsymbol{\lambda}(t_f) = \frac{\partial \Phi}{\partial \mathbf{y}}\Big|_{t_f} \end{aligned} \tag{3.6}$$

Hence, if any of the boundary conditions is a free parameter, either on time or state variables, the above conditions complete the minimum required number of known conditions at the initial or final time. In the matter of low-thrust, final conditions could be unspecified (but possibly still constrained) in many non-rendezvous trajectories such as orbit raising or decreasing, orbital escape problems, gravity assists and so on. Up to this point, the process defined the necessary conditions for a solution to be a stationary one. The *Legendre-Clebsch* condition about local convexity of the *Hamiltonian* shall be satisfied to ensure that the solution is an actual local minimum:

$$\frac{\partial^2 H}{\partial \mathbf{u}^2}\Big|_{\mathbf{u}^*} \geq 0 \tag{3.7}$$

The *TPBVP* defined by Equations (3.5), coupled with the conditions (3.6) and (3.7), has no analytical closed-form solution for complex problems like low-thrust trajectories. Hence, numerical methods shall be employed. However, further information can be obtained by exploitation of the problem's first integrals. If the functions $L$ and $\mathbf{f}$ defined in the System (3.2) do not depend explicitly on the independent variable $t$, then the *Hamiltonian* is a first integral of the TPBVP along an optimal trajectory [2]. In general, if a first integral is found, the redundant information that it generates can be exploited to eliminate one *adjoint equation*, formally transforming the original TPBVP into another one of lower dimension, by following the procedure shown by [11].

### 3.1.2. DIRECT METHODS

A direct method does not require the derivation of the necessary conditions needed by indirect methods. On the contrary, it aims to find a sequence of profiles which progressively reduce the non-augmented performance index $J$ and the constraint's violation. Direct methods require a parameterization of the control functional form over trajectory's arcs. This is generally achieved by two conceptually different methods [2]:

- A grid at different times where the control parameters are to be found and the values within an interval are computed through interpolation.
- A set of orthogonal basis of mathematical functions dependent on time. Usually Fourier series, Legendre polynomials or the Chebyshev ones.

The goal is then to determine the values of the specified free parameters, either control values at fixed times in the grid form or the coefficients of the series in the second case, able to minimize the objective index and to respect the constraints. In this passage, the number of free parameters is reduced from infinite degrees of freedom to a finite number of parameters, depending on the chosen parametrization. This passage could seem a limitation of the direct methods when compared to the indirect ones. However, as already stated in the previous section, a numerical procedure is necessary also for indirect methods when dealing with complex cases such as low-thrust trajectory optimization. These numerical methods require a so-called transcription to convert the infinite-dimension optimal problem into a solvable finite-dimension one. Hence, what seemed a limitation of the direct methods is a required passage of any technique nonetheless.

A direct method's solution is generally not an optimal solution itself, i.e. not a local minimum of the performance index, but just an approximation as a consequence of the discretization or interpolation steps. Hence, the necessary conditions (3.5) and (3.7) can be used as an indicator of how close the found solution is to the real local optimum [14].

### 3.1.3. COMPARISON OF DIRECT AND INDIRECT METHODS

Loosely comparing an optimal control problem to a static constrained optimization, the direct method's goal is to pinpoint a local minimum of the performance function, while an indirect method aims to find a root of the necessary conditions. The latter shall be preferred when a closed-form solution is aimed for. Indeed, indirect methods allow to extract the control in an analytical way [15] [16]. However, this is possible only when several approximations are employed or simplified cases are considered. When a complete low-thrust trajectory problem in a perturbed environment is stud-

ied, a numerical approach is mandatory. In the latter case, a direct method results to be the simpler choice due to several considerations [3]:

- The quantities $\left[\frac{\partial H}{\partial \mathbf{x}}\right]^T$ and $\left[\frac{\partial H}{\partial \mathbf{u}}\right]^T$ needed by indirect methods must be analytically computed and changed when different models are employed, e.g. environmental perturbations. Furthermore, when a problem is divided into phases, e.g. a patched-conic approach for an interplanetary trajectory, these quantities change along the trajectory. This requires an extensive preliminary analytical stage for any different problem in the matter. On the contrary, a direct method is a flexible approach, more suitable for *black box* implementations, and able to handle a problem divided into different phases;

- Path inequalities, which are quite ordinary in low-thrust applications, represent a relevant issue for indirect methods. Indeed, a first guess of the *active-inactive* sequence is needed for practical methods as it changes the form of the Hamiltonian, by adding the *Lagrange multipliers*, the number of constrained arcs and the junction conditions. However, an a-priori knowledge of the right series is quite hard to achieve;

- Another issue with first guesses emerges from the initial estimate of the *adjoint variables* $\boldsymbol{\lambda}$. As remarked by [17], the extremal solutions can be very sensitive to small changes in the unspecified boundary conditions. As usually the initial state variables are specified, the *transversality conditions* (3.6) show that the initial values of the adjoint variables for the optimal trajectory are not known. Further, these variables are not representing physical quantities. Hence, setting the right initial conditions, or even reasonable ones, is very complex, and a bad initialization often results in numerically ill-conditioned solutions. On the contrary, direct methods disregard those variables and require only initial guesses on the physical state and control variables.

## 3.2. PRACTICAL TECHNIQUES FOR OPTIMAL CONTROL

As stated in numerous occasions, in general the continuous optimal control problem does not have a closed-form solution and practical numerical optimization methods come into play. Any numerical technique cannot handle an infinite-dimension problem, but it needs a discrete problem with a finite set of variables and constraints to work with. This transition can be performed with conceptually different methods which will be investigated in the present section. It is important to emphasize that the following techniques are applicable to both indirect and direct approaches. However, as the current thesis work will revolve on a direct method, a major focus will be paid to the latter formulation. This choice is further justified by the considerations made in Section 3.1.3. A complete review of the common methods for low-thrust trajectory optimization has been compiled by [18], whereas in this section two major classes will be addressed.

### 3.2.1. SINGLE SHOOTING

Typically, the single-shooting method does not actually find application in the field of low-thrust optimal control. However, it is useful to introduce the notation and several concepts shared by its extension, the *Multiple Shooting* method. The discretization grid is composed by only two points, the initial and final times. Initially, the $n$ free parameters in $\mathbf{y}^T = [\bar{x}_1, \ldots, \bar{u}_{n_c}]$, composed by the initial conditions and the control parameters, are guessed. Hence, the trajectory is propagated forward (or equivalently backward) from the starting to the end time, leading to the final state:

$$\mathbf{x}_f^p = \mathbf{x}_0 + \int_{t_0}^{t_f} \mathbf{f}(t, \mathbf{x}, \mathbf{u})\, dt$$

In general the propagated state $\mathbf{x}_f^p$ will not coincide with the required final one $\mathbf{x}_F$. Hence, the difference between these two quantities becomes a constraint to nullify. In literature, this constraint is generally labeled as *defect*:

$$\mathbf{c}(\mathbf{y}) = \mathbf{x}_f^p - \mathbf{x}_f \tag{3.8}$$

The numerical values of the violation of the boundary conditions can be exploited to iteratively adjust the control parameters with the NLP algorithms presented in Section 3.3, in order to finally solve the constrained minimization.

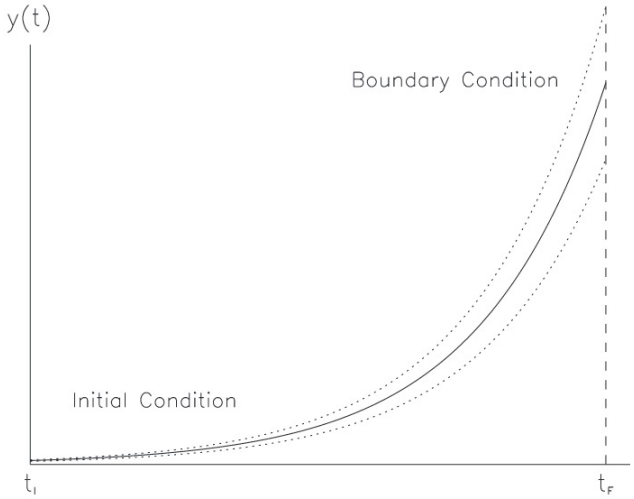



Figure 3.1: Single shooting method illustration [3].

The advantage of this basic method is that the NLP sub-problem has only a small number of variables to optimize, i.e. the initial state guess and control parameters. However, due to the usual long time-span of a low-thrust maneuvers, even small changes in the parameters can result in very large defects change, leading to hypersensitivity with respect to the free parameters.

### 3.2.2. MULTIPLE SHOOTING

In order to overcome the drawback of parameter sensitivity, it is possible to segment the overall time interval into a set of $m-1$ smaller steps discretizing the interval at $m$ grid points $t_0 < t_1 < t_2 < \cdots < t_f$. Then, each of the segments can be treated as an independent single-shooting method, with continuity constraints added. Therefore, first guesses of the $n_s$ state variables for each intermediate segment are now needed. The first guess trajectory is usually found by fast and low-fidelity methods, such as the shape-based techniques seen in Section 2.3 for space applications. The state variables at intermediate grid points are now control variables to be optimized. Hence, the number of control parameters in $\mathbf{y}$ increases with respect to the single shooting method, precisely $n_y = (m-1)(n_s + n_c \cdot n_p)$, where $n_s$ is the number of state variables, $n_c$ the control components and $n_p$ the control parameters per each component. The *defect* equations can be expressed in the general form as:

$$\mathbf{c}(\mathbf{y}) = \begin{pmatrix} \mathbf{x}_2^p - \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_f^p - \mathbf{x}_f \end{pmatrix} \tag{3.9}$$

where again the goal is to nullify $\mathbf{c}(\mathbf{y})$.

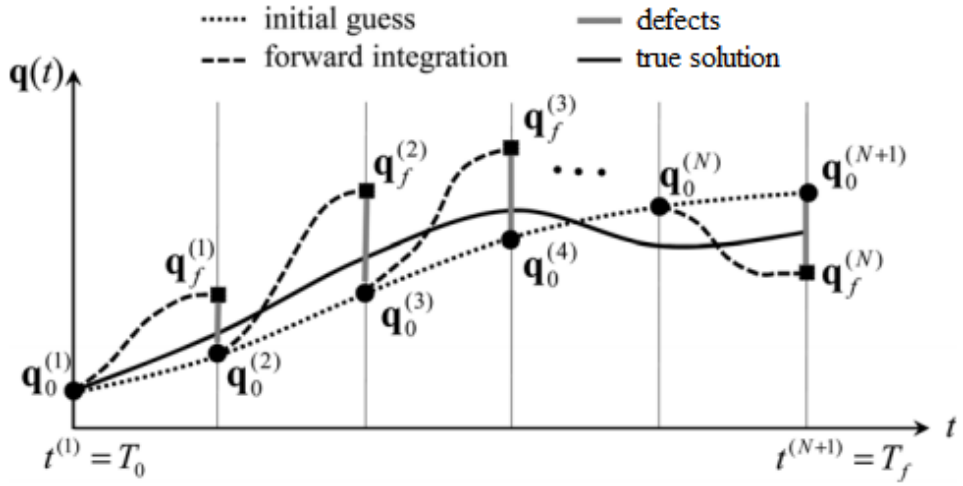



Figure 3.2: Multiple shooting method illustration [19].

The dimension that the NLP sub-problem shall solve, in order to link the different phases and minimize the objective function, dramatically increases with increasing number of steps. However, the effects of changing a particular parameter are more intuitive for smaller steps, leading to an improvement of the convergence properties. In addition, the drawback of the single shooting is solved as, when the number of steps is high enough, the variables in the first stages of the trajectory do not influence the last phases. The segment decoupling mathematically translates into very sparse *Jacobian* and *Hessian* matrices, later involved by the NLP algorithm. For example, the Jacobian gets sparser and sparser as more phases are employed, because the percentage of non-zero elements is proportional to $1/(m-1)$. This sparsity can be exploited to construct a computationally efficient nonlinear programming subroutine, making the multiple shooting method both robust and competitive. While the general concept of a multiple shooting transcription scheme has been presented in this section to introduce the discussion on the method selection of Section 3.2.4, it will be better explained and examined in detail in Section 5.1 as it is the central topic of the present thesis.

### 3.2.3. COLLOCATION

The basic goal of *collocation* methods is to avoid repeated propagations over each segment. This is achieved by partitioning again the whole trajectory into $m-1$ segments, leading to $m$ grid points. Hence, the trajectory is only represented by the set of state variables $\mathbf{x}(t_k)$ and their derivatives $\mathbf{f}(t_k, \mathbf{x}(t_k), \mathbf{u}(t_k))$ at mesh points as well as the control profile nodes $\mathbf{u}(t_k)$. As these values are treated as NLP variables, gathered in the vector $\mathbf{y}$, the optimal control problem has been completely transcribed into a finite-dimensional NLP. For this reason, also collocation methods need a first-guess solution, which can be sought with the aforementioned approaches. The state, state-derivative and control values within each interval are computed by interpolation through piecewise functions, usually Hermite (third order), Chebyshev or Lagrange polynomials (see [3] for detailed schemes) or Fourier series [20], whose coefficients depend on the adjacent grid points' state and derivatives. This a-priori shape replaces the numerical integration process of shooting techniques with a much faster analytical propagation.

The differential equations $\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t))$ are substituted by a discretized form, which for a simple Euler scheme takes the following form:

$$\dot{\mathbf{x}} = \mathbf{f}(t_k, \mathbf{x}_k, \mathbf{u}_k) \approx \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{h} \tag{3.10}$$

where $h$ is the interval size $t_{k+1} - t_k$. This Euler form is then transformed into a set of NLP constraints to be nullified:

$$c_k(\mathbf{y}) = \|\mathbf{x}_{k+1} - \mathbf{x}_k - h\mathbf{f}(t_k, \mathbf{y}_k, \mathbf{u}_k)\| \tag{3.11}$$

These constraints, which ensure the equation of motion to be approximately satisfied, are then coupled with the fixed-event ones, to construct a continuous trajectory, and the path constraints, to respect the requested bounds at the grid points.

In collocation methods the choice of the interval size is vital because it influences the accuracy of the interpolated function in representing the true trajectory. An efficient procedure could be to compute initial estimates with a sparse grid and then refine it progressively. This implementation makes this technique very robust to imprecise initial guesses. Also in this method, the sparsity of the matrix shall be exploited as much as possible to make the algorithm efficient.

The greater drawback of collocation methods is that for problems dominated by highly nonlinear dynamics, a very dense grid is needed to compute an accurate solution which, when integrated forward for validation, leads to small errors in the final state. This problem arises from the finite-difference approximation of the dynamics, as in Equation (3.10) for an Euler scheme, and from the parametrization of the shape. However, a dense grid translates into a huge matrix inversion during the NLP sub-problem (see Section 3.3), leading to a degradation of the computational performance.

### 3.2.4. METHOD SELECTION

Generally the method selection is driven primarily by the nature of the problem in the matter. For example, if the functional form of the state variables evolution is known it is more advisable to choose a collocation method employing that particular functional form as interpolation function. On the contrary, when the state evolution is unpredictable, multiple shooting methods ensure that the correct functional form is achieved by appropriate numerical propagation, leading to more accurate solutions.

Computational efficiency is another driving factor. From this point of view, the analytical propagation makes the collocation methods much faster than the multiple shooting techniques. However, as already explained in the previous section, when an accurate solution is sought, this difference is reduced, if not reversed at all, due to the much denser grid needed by collocation methods. Indeed, in practical applications, collocation methods could reach a limit in the final accuracy, when the computed trajectory is integrated for validation, which cannot be improved if not with dramatically dense grids which lead to unrealistic computational times. On the contrary, multiple shooting techniques become more accurate as the grid becomes progressively denser.

For the aforementioned reasons, the multiple shooting method seems the best compromise between attractive computational times and very accurate solutions. On the other hand, the collocation method can be employed as a very efficient and quite accurate method to compute first guesses [20]. Indeed, for deep space missions, also [2] suggested as a good hybrid solution to start the solution process with a collocation method employing a sparse grid, then to progressively make it denser in order to improve the candidate solution accuracy, and finally refine the guess with a multiple-shooting technique.

## 3.3. NONLINEAR PROGRAMMING

In the preceding sections it has been emphasized how an optimal control problem shall be converted to a finite-dimensional constrained optimization to be actually solved by selecting the opti-

mal combination of the control variables. This is achieved by an iterative process able to adjust the optimizable parameters as function of the objective function value and constraints' violation at the previous iteration. *Nonlinear programming* (NLP) is a method, based on Newton's secant method, able to calculate the control increments allowing both quantities to be reduced.

Generally stated, the NLP problem aims to find the $n$-dimensional vector $\mathbf{y}$ such that the objective function $F$ is minimized:

$$\min F(\mathbf{y}), \quad \mathbf{y} \in \mathbb{Y} \subseteq \mathbb{R}^n \tag{3.12}$$

while respecting $m \leq n$ equality constraints and $l$ inequality constraints:

$$\begin{aligned} \mathbf{a}(\mathbf{y}) &= \mathbf{0}, \quad \mathbf{a} \in \mathbb{R}^m \\ \mathbf{b}(\mathbf{y}) &\geq \mathbf{0}, \quad \mathbf{b} \in \mathbb{R}^l \end{aligned} \tag{3.13}$$

For the purpose of procedure's clarity, first the case with only equality constraints will be addressed. Then, in order to learn how to handle all the constraints individually, the case with only inequality ones is presented. Hence, the method will be generalized to treat both constraints at the same time within the example framework of *quadratic programming*. This reasoning is carried out to introduce two algorithms for solving the general nonlinear problem, i.e. the *sequential quadratic programming* (SQP) approach and the *interior point* (IP) technique. WORHP (Worhp Optimises Really Huge Problems), the NLP solver interfaced with the multiple shooting tool, utilizes a combination of SQP and IP, as shortly outlined in the last part of the section.

### 3.3.1. EQUALITY CONSTRAINTS NLP

The condition in Equation (3.13)-1 on the number of independent equality constraints $m$ to be less than or equal to the number of free parameters $n$ is required to guarantee enough degrees of freedom in the selection of the solution. Otherwise, the space of admissible solutions $\mathbb{X}$ will be empty.

The classical procedure is to construct the *Lagrangian* adjoining the equality constraints with the objective function:

$$\mathscr{L}(\mathbf{y}, \boldsymbol{\lambda}) = F(\mathbf{y}) + \boldsymbol{\lambda}^T \mathbf{a}(\mathbf{y}) \tag{3.14}$$

where the $\lambda_k$ coefficients are again known as *Lagrange multipliers*. From this step on, the problem is formulated as an unconstrained optimization. The first-order necessary conditions for a local optimum $(\mathbf{y}^*, \boldsymbol{\lambda}^*)$ require the state-derivatives of the Lagrangian to be zero:

$$\begin{aligned} \nabla_y \mathscr{L}(\mathbf{y}^*, \boldsymbol{\lambda}^*) &= \mathbf{0} \\ \nabla_\lambda \mathscr{L}(\mathbf{y}^*, \boldsymbol{\lambda}^*) &= \mathbf{0} \end{aligned} \tag{3.15}$$

The unconstrained second-order necessary condition requires the curvature of the Lagrangian to be positive along any perturbation $\delta\mathbf{x}$ with respect to the minimum. However, in the constrained case not any perturbation direction is always feasible, i.e. there are increment directions which would violate the constraint's satisfaction. Hence, we shall require a positive curvature along any allowed perturbation direction $\delta\mathbf{v}$ which lies in the constraint tangent space, i.e. those ones which respect the constraints. Defining the *Hessian* as:

$$\mathbf{H}_{\mathscr{L}} = \nabla^2_{yy} \mathscr{L} \tag{3.16}$$

the necessary condition for a minimum become:

$$\delta\mathbf{v}^T \mathbf{H}_{\mathscr{L}} \delta\mathbf{v} \geq 0 \tag{3.17}$$

while the sufficient condition just substitutes the ≥ operator with the > one. However, to check directly the second-order necessary condition in this form is non-trivial. On the other hand, methods to assess if a matrix is positive definite for any perturbation direction have been extensively studied and general procedures exist. [11] explains how to convert the condition 3.17 on the Hessian to the study of (semi)positive definiteness of another counterpart matrix. The constraint tangent subspace has an $(n-m)$ basis for the null-space. The feasible perturbations $\delta\mathbf{v}$ can therefore be described as a linear combination of these basis. If they are gathered into a matrix $Y$ as columns, it is possible to write:

$$\delta\mathbf{v} = Y\boldsymbol{\eta} \tag{3.18}$$

where $\eta$ is a vector of coefficients. Now, the second-order necessary condition (or equivalently the sufficient one) can be restated as:

$$\boldsymbol{\eta}^T \left[ Y^T \mathbf{H}_{\mathscr{L}} Y \right] \boldsymbol{\eta} \geq 0 \tag{3.19}$$

which is simply satisfied if the *projected Hessian* $Y^T\mathbf{H}_{\mathscr{L}}Y$ is (semi)positive definite.

### NEWTON METHOD

This method falls under the general class of *Gradient techniques*, which use a polynomial expansion of the objective function and the contraints. Indeed, NLP algorithms rely on a Taylor-series expansion at the second-order for the fitness function in Equation (3.12), and at the first-order for the equality constraints in Equation (3.13)-1, in order to practically find a solution able to satisfy the necessary conditions (3.15) and (3.19). The second-order series is necessary for the performance function because linear functions have no local optima:

$$\begin{aligned} F(\mathbf{y}) &= F(\mathbf{y_v}) + \mathbf{g}^T(\mathbf{y_v})(\mathbf{y}-\mathbf{y_v}) + \frac{1}{2}(\mathbf{y}-\mathbf{y_v})^T\mathbf{H}(\mathbf{y_v})(\mathbf{y}-\mathbf{y_v}) + O[(\mathbf{y}-\mathbf{y_v})^3] \\ \mathbf{a}(\mathbf{y}) &= \mathbf{a}(\mathbf{y_v}) + \mathbf{G}^T(\mathbf{y}-\mathbf{y_v}) + O[(\mathbf{y}-\mathbf{y_v})^2] \end{aligned} \tag{3.20}$$

where $\mathbf{g}^T = \nabla_y F$ and $\mathbf{H} = \nabla^2_{yy} F$ are respectively the *gradient* and the *Hessian* of the objective function, while $\mathbf{G} = \partial\mathbf{a}/\partial\mathbf{y}$ is the *Jacobian* of the constraints. Substituting the truncated expansions in the necessary conditions (3.15), followed by simple mathematical rearrangement in a matrix form, leads to the linear system:

$$\begin{pmatrix} \mathbf{H} & \mathbf{G}^T \\ \mathbf{G} & \mathbf{0} \end{pmatrix}_{\mathbf{y_v}} \begin{bmatrix} \mathbf{y}-\mathbf{y_v} \\ \boldsymbol{\lambda} \end{bmatrix} = -\begin{pmatrix} \mathbf{g} \\ \mathbf{a} \end{pmatrix}_{\mathbf{y_v}} \tag{3.21}$$

These two equations are known as *Karush-Kuhn-Tucker* system (KKT). It defines the search direction $\mathbf{p} = \mathbf{y}-\mathbf{y_v}$ from the previous iteration and the new value of the Lagrange multipliers $\boldsymbol{\lambda}$, able to locally reduce the objective function value and the constraints violation. The new candidate solution is then $(\mathbf{y}, \boldsymbol{\lambda})$ where $\mathbf{y} = \mathbf{y_v} + \alpha\mathbf{p}$ with $0 < \alpha \leq 1$. In line-search methods the step-size $\alpha$ is iteratively adjusted to minimize the objective function and constraints' violation.

### IMPLEMENTATION OF NEWTON METHOD

The KKT system requires computation of the gradient $\mathbf{g}$, the Jacobian matrix $\mathbf{G}$ and the Hessian matrix $\mathbf{H}$. Direct methods for general applications compute these quantities by finite-difference approximations. The trajectory is evaluated again with perturbations in each of the control parameters and then numerical differencing is applied. For instance, the contents of the k-column of the Jacobian matrix can be computed with a central difference method as:

$$\mathbf{G}(:,k) = \frac{\mathbf{c}(\mathbf{y}+\delta_k \mathbf{e}_k) - \mathbf{c}(\mathbf{y}-\delta_\mathbf{k}\mathbf{e}_\mathbf{k})}{2\delta_k} \tag{3.22}$$

where $\mathbf{e}_k$ is the optimal unit vector for k-column defined direction and $\delta_k$ the perturbation in that direction (see [3] for full mathematical formulation). In principle, this step requires the integration of the equations of motion for each perturbation at every iteration. Although this is a general approach working for every problem submitted to the algorithm, this passage turns out to be the bottleneck of the whole process both in terms of computational speed, as gradient calculations for multiple-shooting takes around 70 % of execution time [21], and derivative accuracy.

Since the goal of the thesis is to create an efficient tool for high-fidelity low-thrust trajectory optimization, this general approach will be replaced by an approach based on the integration of the so-called *variational equations*. This new set of equations, which describe the exact evolution of first and second-order control and state derivatives of the NLP variables (state, constraints, etc.), is propagated as well during one phase granting lower computational times, as the number of iterations is dramatically reduced, and a more precise computation of the gradient, Jacobian and Hessian quantities for NLP iterations [2]. Indeed, the derivatives' accuracy will now only depend on the chosen integrator and step-size because no other approximation is made in their computation. The variational equations are problem-dependent and thus the generality can be compromised.

A detailed analysis of both approaches will be presented in Chapter 4, while the implementation details of the variational approach will be discussed in Section 5.1.

### 3.3.2. INEQUALITY CONSTRAINTS NLP

The number $l$ of inequality constraints can be greater than the number $n$ of control parameters because these constraints remove degrees of freedom only when they are *active*. An inequality constraint is labeled as *active* when it is satisfied as an equality, while if it is respected as a pure inequality, it is named *inactive*. The *Lagrangian* is similarly defined as:

$$\mathscr{L}(\mathbf{y},\boldsymbol{\lambda}) = F(\mathbf{y}) + \boldsymbol{\mu}^T \mathbf{b}(\mathbf{y}) \tag{3.23}$$

The necessary conditions are still the same as defined for the purely constrained problem. However, a further condition on inequality *Lagrange multipliers* shall be satisfied at a local optimum:

$$\begin{cases} \mu_i < 0 & if \;\; b_i \; active \\ \mu_i = 0 & if \;\; b_i \; inactive \end{cases} \tag{3.24}$$

Following the notation of [3], at a generic point $\mathbf{y}$ the totality of active constraints is called the *active set*, while the inactive constraints together form the *inactive set*. Clearly, once we know the set of active constraints, the problem can be handled with the techniques seen in the previous section. Thus, an *active set strategy* is needed to pinpoint which constraints are active and which are not. An example will be given in the next section about the general *quadratic programming* formulation.

### 3.3.3. QUADRATIC PROGRAMMING

A *quadratic programming* problem is a particular form of the general NLP method with a quadratic objective function and linear constraints.

$$\begin{aligned} \min F(\mathbf{y}) &= \mathbf{g}^T\mathbf{y} + \frac{1}{2}\mathbf{y}^T\mathbf{H}\mathbf{y} \\ \mathbf{A}\mathbf{y} &= \mathbf{a} \\ \mathbf{B}\mathbf{y} &\geq \mathbf{b} \end{aligned} \tag{3.25}$$

Even if it is a simplified formulation, this method is exploited to illustrate the role of *active set strategy*. Indeed, as the KKT system has been obtained by quadratic and linear approximations of the objective and the constraints respectively, the *Newton method* would only need one iteration when the right set of active constraints is selected. However, this is not a trivial problem as usually no precise a-priori information is available and thus an *active set strategy* is required. The procedure can be summarized as:

- First guesses of point $\mathbf{y_v}$ and active set $\mathbb{B}_v$ are given.

  Now, $\tilde{\mathbf{b}} \subset \mathbf{b}$ is the subset corresponding to active constraints, with $\tilde{\mathbf{B}}$ the respective Jacobian matrix.

- Solve the *Karush-Kuhn-Tucker* system considering the active set constraints as equalities and neglecting the inactive ones. The KKT system can be re-arranged as follows [3]:

$$\begin{pmatrix} \mathbf{H} & \mathbf{A}^T & \tilde{\mathbf{B}} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \tilde{\mathbf{B}} & \mathbf{0} & \mathbf{0} \end{pmatrix}_{\mathbf{y_v}} \begin{bmatrix} \mathbf{y}-\mathbf{y_v} \\ \boldsymbol{\lambda} \\ \boldsymbol{\mu} \end{bmatrix} = - \begin{pmatrix} \mathbf{g} \\ \mathbf{a} \\ \tilde{\mathbf{b}} \end{pmatrix}_{\mathbf{y_v}} \tag{3.26}$$

- Increment of the control vector.

  Using the computed direction the new point results from $\mathbf{y} = \mathbf{y_v} + \alpha(\mathbf{y}-\mathbf{y_v})$, with the greater $0 \leq \alpha \leq 1$ not violating any inequalities.

- Check on the step-length.

  By selecting the largest possible step not violating any inactive constraint, the *active set strategy* comes into play and two alternatives are possible:

  - if $\alpha < 1$ it means that the new point is on the "border" of an inequality constraint that was before considered as inactive. Hence the new constraint is included in the active set $\mathbb{B}_v$, the quantities $\tilde{\mathbf{b}}$ and $\tilde{\mathbf{B}}$ updated and the algorithm reiterates.
  - if $\alpha = 1$ and the conditions (3.24) are fulfilled, the optimum of the quadratic problem is found, otherwise the equalities with positive multipliers shall be removed from the active set and the algorithm reiterates.

This simple quadratic example illustrates just one of the possible active set strategies, on which conceptual modifications in any step of the process are possible. Nonetheless, it is a clear illustration of the NLP process and the set of strengths and difficulties that this method implies. Detailed discussions and examples can be found in [22].

### 3.3.4. SEQUENTIAL QUADRATIC PROGRAMMING

The Sequential Quadratic Programming (SQP) method is a widely used approach to solve general nonlinear problems as in Equations (3.12) and (3.13). The basic algorithm is constructed as a descent method employing line search, based on the following iterative steps once started at a given initial point $\mathbf{y}_v$ [3]:

1. Check termination criteria for current point $\mathbf{y}_v$. If criteria are satisfied, exit the iteration loop, otherwise continue;
2. Approximate the nonlinear problem by a quadratic subproblem around $\mathbf{y}_v$;

3. Find a search direction **p** through a quadratic programming algorithm (see Section 3.3.3);
4. Determine the stepsize by means of a line search method;
5. Update the current point and return to point 1.

For a detailed discussion on SQP algorithms the references [23] and [24] can be consulted.

### 3.3.5. INTERIOR POINT

Another family of algorithms to solve nonlinear constrained optimization problems involves penalty functions and it is labeled as *penalty-barrier* or *interior point* (IP). The basic idea is to translate the constrained problem into an equivalent unconstrained optimization one. The new objective function can be written as [25]:

$$\beta = F(\mathbf{y}) - \mu \sum_i \ln b_i(\mathbf{y}) + \frac{1}{2\mu} \sum_j \| a_j(\mathbf{y}) \| \tag{3.27}$$

where $\mu$ is a scaling factor, known as barrier parameter. The last term describes the penalty factor for the equality constraints' violation, while the logarithmic term handles the inequality constraints. The latter becomes bigger when $b_i$ tends to zero, i.e. when the inequality constraint becomes active. Therefore, the minimization routine of the new objective function will try to move away from the barriers where the inequality constraints are active. Exhaustive information and several alternative forms can be found in the extended review [25].

### 3.3.6. WORHP: EUROPEAN NLP SOLVER

WORHP [26] is a NLP solver designed to efficiently solve generic nonlinear constrained optimization problems, spanning from small- to large-scale. It combines a SQP algorithm for the general nonlinear level with an primal-dual IP method for its associated quadratic subproblem. The primal-dual IP is an alternative of the basic IP algorithm, whose details, which are beyond the scope of the current thesis, can be found in [25]. This method combination is used to find a series of search directions which are scaled using a line search with the Augmented Lagrangian merit function.

The necessary derivative information can be automatically computed by finite-difference or quasi-Newton updates (see Section 4.1), or supplied by the user, as it will be done in the current thesis with the variational approach. WORHP is suitable for large-scale problems as it intrinsically works with sparse matrices for computational performance and memory consumption reasons. Thanks to this feature, the problem size is limited only by the working machine memory. Indeed, WORHP performed efficiently on a variety of huge-dimension problems [27]. A more detailed discussion on the sparsity patterns generated by multiple shooting schemes and the way WORHP handles sparse matrices is carried out in Section 5.1, in particular concerning practical implementation details.

## 3.4. GENERAL OPTIMIZATION

All of the methods discussed so far in this chapter need a first-guess solution close enough to the global optimum as they all only behave as local optimizers for the selection of the control parameters, after that some design variables had already been selected. For example, also basic rendezvous missions with no flybys usually present multi-modality with respect to the departure date with a periodicity given by the synodic period of the departure and arrival planets, as depicted in Figure 3.3.

A local optimizer, even if using also departure date and time of flight as free parameters, cannot find the real minimum if the local algorithm starts from a poor initial guess.

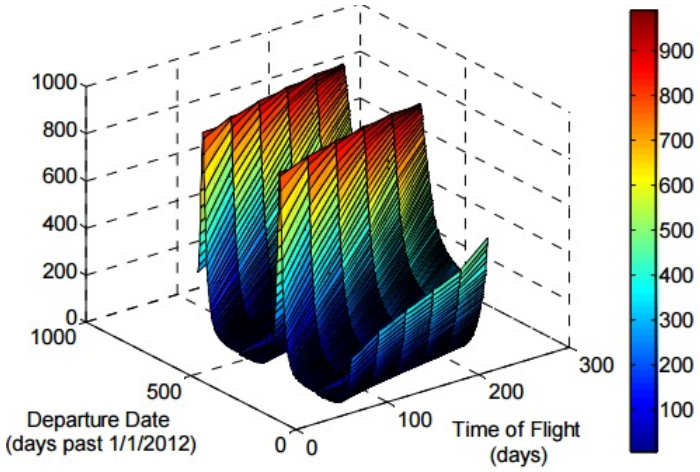



Figure 3.3: Illustration of typical multi-modal behavior of final spacecraft mass [1].

To quickly span the complete domain of possible combinations between the *global variables* (e.g. departure date, time of flight, number of revolutions, selection and duration of flybys, et cetera) computationally efficient techniques shall be employed in combination with global search methods. Indeed, as repeatedly stated, shape-based approaches or semi-analytical approximations have proven to be computationally fast in assessing the physical characteristics of a low/medium-fidelity low-thrust rendezvous. However, typically they are not able to describe maneuvers different from the thrusted two-body problem. In a more realistic scenario, those methods are used to describe the thrusted phases, e.g. between two successive gravity-assists, and they are then coupled with a global optimizer able to handle the *global variables* selection [28] [29] [30], which will be now discussed.

### 3.4.1. SAMPLING METHODS

A sampling method simply picks values for the parameters' vector within the complete domain of possible combinations of the *global variables* following a prescribed procedure [31]. Because the whole domain is usually too wide to be completely investigated, sampling methods aim to characterize different areas' performance by selecting only a few points inside them, and rely on the idea that points near the investigated one would show similar characteristics. The main goal of this family of techniques in low-thrust trajectories is to reduce the search space selecting the areas of greater interest where to utilize smarter (see Section 3.4.2) or local optimizers.

This family of methods usually finds application in the optimization of the *static parameters* involved in the low-thrust trajectory problem. Static parameters are those non-dynamic variables that are constant along a trajectory and part of the aforementioned *global variables*. Usually, examples of this parameters' family are the departure date, the time of flight (or equivalently the arrival date), the number of complete revolutions. Even the number of gravity-assists and their sequence belong to this category but are rather optimized with Heuristic algorithms that will be explained in the next section.

#### UNIFORM GRID SEARCH

Suppose that the parameters to be optimized are gathered in the vector $\mathbf{x}$, constrained to be within given bounds $[\mathbf{x_L}, \mathbf{x_U}]$, which define the search space. For each parameter a step-size $h_j$ is se-

lected and the interval $[x_L, x_U]_j$ in which every parameter is bounded is divided into $n_j = \frac{(x_U - x_L)_j}{h_j}$ uniform intervals with $n_j + 1$ grid points [31]. Hence, the total search space is discretized with $(n_1+1)(n_2+1)\dots(n_j+1)\dots$ grid points. When three parameters are employed, the search grid can be visualized as a rectangular parallelepiped with a vertex of coordinates $[x_{L_1}, x_{L_2}, x_{L_3}]$ and the opposite one $[x_{U_1}, x_{U_2}, x_{U_3}]$, whose sides are representing the intervals $[x_L, x_U]_j$.

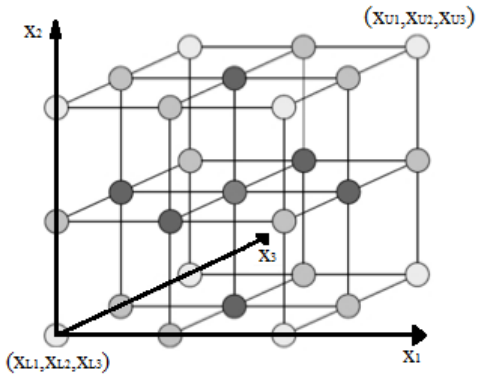



Figure 3.4: Illustration of a uniform spaced grid.

Usually this kind of search space discretization is employed to find convenient combinations of departure dates and time of flight. Those parameters influence the relative phasing of the departure and arrival orbits and are suitable for this type of analysis because a small change in the candidate vector shall not change significantly the corresponding performance index value. Also the number of complete revolutions is usually spanned with a uniform grid as this parameter can only assume positive integral values, and is thus very suitable for it.

### MONTECARLO METHOD

MonteCarlo is the purest form of random sampling. The algorithm generates a series of uncorrelated sampling vectors **x** within the domain of which the corresponding performance output will be computed. As any other sampling method, its power is the ability to work regardless of the objective function derivatives. MonteCarlo analysis is not a robust method when the number of samples is low as it relies on a random population. Unfortunately, there is not a general method to assess a robust minimum number of samples but it shall be found by trial and error. This can be translated into a great and inefficient computational effort. Possible alternatives are the so-called *quasi MonteCarlo* methods, which represent the compromise between pure random and systematic approaches. A detailed analysis of basic and advanced methods can be found in the book by [31].

### QUASI MONTECARLO

Intermediate techniques generating near-random samples to cover a multi-dimensional distribution fall under the denomination of Quasi MonteCarlo methods. The samples can be generated through low-discrepancy sequences like Sobol's or Faure's, which try to cover the less filled regions. In detail, the samples, also called Low Discrepancy Points, are generated in the attempt to cleverly, and not only randomly, fill an hypercube with unit hyper-edges. Latin hypercube sampling (LHS) is a statistical method gathered in this general denomination, which on average yields slightly better results than the pure MonteCarlo technique.

### 3.4.2. HEURISTIC METHODS

Heuristic methods (HM) are a quite modern class of algorithms created to improve the robustness, efficiency and multi-modality characteristics of a static optimization problem when compared to traditional methods [1]. One advantage, as for sampling methods, is that they work regardless of any knowledge of the function derivatives. A further one is the ability to deal with multi-objective optimization. These methods have found application in the low-thrust case in the optimization of the free parameters of shape-based methods [29], discussed in Section 2.3, or in finding good combinations of coast arcs and thrusted phases with semi-analytical methods [20].

A huge realm of conceptually different algorithms lies under the denomination of heuristic methods, but there are some common passages and characteristics. HMs start with a variably random generated population of global variables and iteratively produce new candidate points by following a fixed evolution/generation scheme (usually mimicking some natural, social or scientific behavior), based only on evaluation of the objective function outputs at previous iterate(s).

Among the infinite number of different algorithms, *evolution* based methods are the most famous and commonly implemented heuristic methods [31]. On the other hand, *ant colony* methods are those ones that have introduced peculiar, interesting and innovative principles [32], and which are then reproduced with several conceptual changes in other variants, e.g. *particle swarm* etc.

## 3.5. CONCLUSIONS

This chapter discussed the key background topic of the current thesis work. The general optimal control theory has been presented in close contact with the practical methods to solve it. After a survey of direct and indirect methods' strengths and weaknesses, a comparison has explained why a direct approach is more advisable for complex non-integrable problems and thus will be employed for the tool development.

Subsequently, different techniques have been presented and again a selection has been conducted looking for the best compromise between computational time and solution accuracy. This leads to the selection of a multiple shooting approach. The bottleneck of common methods has been identified and a practical method based on variational equations to overcome it has been introduced. The background theory, given by the mathematical branch of calculus of variations, will be presented in the next chapter. On the other hand, it has been explained how collocation can be possibly employed for first estimates or hybrid methods.

In the middle part of the chapter, NLP fundamentals to solve the discretized optimization subproblem have been examined. Both SQP and IP algorithms' bases have been outlined, as a combination of them is used in WORHP, the NLP solver which will be interfaced with the novel developed tool.

Finally, general optimization methods have been presented with particular care on what application they can find in the low-thrust case. Sampling techniques are used in the research to treat static parameters as departure date, time of flight and number of revolutions, while heuristic methods can be executed to find good combinations of coast and thrust arcs, as well as convenient flybys sequences in complex test cases.

# 4

# DERIVATIVE COMPUTATION

All local optimization algorithms, including the approach used by WORHP, require derivative information to investigate the hyper-dimensional neighbourhood of the nominal point in terms of objective function improvement and constraint satisfaction. As discussed in the previous chapter, three quantities shall be computed:

- Gradient of the objective function J:

$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial p_1} & \frac{\partial J}{\partial p_2} & \cdots & \frac{\partial J}{\partial p_{n_y}} \end{bmatrix}^T \tag{4.1}$$

- Jacobian of the constraints

$$\mathbf{J_c} = \begin{bmatrix} \frac{\partial g_1}{\partial p_1} & \frac{\partial g_1}{\partial p_2} & \cdots & \frac{\partial g_1}{\partial p_{n_y}} \\ \frac{\partial g_2}{\partial p_1} & \frac{\partial g_2}{\partial p_2} & \cdots & \frac{\partial g_2}{\partial p_{n_y}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial g_{m_c}}{\partial p_1} & \frac{\partial g_{m_c}}{\partial p_2} & \cdots & \frac{\partial g_{m_c}}{\partial g_{n_y}} \end{bmatrix} \tag{4.2}$$

- Hessian of the augmented Lagrangian

$$\mathbf{H}_{\mathscr{L}} = \begin{bmatrix} \frac{\partial^2 \mathscr{L}}{\partial p_1^2} & \frac{\partial^2 \mathscr{L}}{\partial p_1 \partial p_2} & \cdots & \frac{\partial^2 \mathscr{L}}{\partial p_1 \partial p_{n_y}} \\ \frac{\partial^2 \mathscr{L}}{\partial p_2 \partial p_1} & \frac{\partial^2 \mathscr{L}}{\partial p_2^2} & \cdots & \frac{\partial^2 \mathscr{L}}{\partial p_2 \partial p_{n_y}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 \mathscr{L}}{\partial p_{n_y} \partial p_1} & \frac{\partial^2 \mathscr{L}}{\partial p_{n_y} \partial p_2} & \cdots & \frac{\partial^2 \mathscr{L}}{\partial p_{n_y}^2} \end{bmatrix} \tag{4.3}$$

where $n_y$ is the number of free parameters $p_j$ in $\mathbf{y}$, and $m_c$ the number of constraints $g$. Both first- and second-order derivatives are needed to compute the new search direction with quadratic-programming step (see Section 3.3.3). Sometimes $\nabla J$, $\mathbf{J_c}$ and $\mathbf{H}_{\mathscr{L}}$ will be labeled respectively as DF, DG and HM, following the nomenclature used in WORHP [26].

The classic approach to approximate the derivatives by finite-differences will be introduced in Section 4.1, describing both second-order finite-difference schemes and quasi-Newton methods to construct this information recursively. These methods, implemented alternatives within WORHP, will be used for internal validation of the supplied derivatives computed by a variational approach, the main novelty of this thesis research, described in detail in Section 4.2. The variational approach to compute the derivative information leads to an increased complexity in the problem setup. However, once the functional form of the partial derivatives has been established, the improvement in computational performance is remarkable. Only the first-order derivatives variational computation leads to an improvement of a factor of 3 to 4 in computational times with respect to finite-differences [2].

In order to avoid futile repetitions, where possible the equations will be derived for a generic scalar function $f$, which can represent the objective function, a scalar constraint or the Lagrangian. In the remainder of the chapter, time, state and control will be denoted as variables when their continuous acceptation is involved, and as parameters when their transcribed and discrete counterparts are intended.

## 4.1. FINITE-DIFFERENCE APPROACH

The matrices in Equations (4.1)-(4.3) are purely composed by objective function or constraint derivatives, or their linear combination through dual variable coefficients (see Equation (3.14)). In order to provide a simpler user-interface and to handle any dynamics, general-purpose optimization tools often approximate these derivatives by finite-difference methods. To understand the procedure in detail, let's consider the Taylor expansion of a one-variable scalar function $f(x)$, infinitely differentiable about a nominal point $\bar{x}$:

$$f(\bar{x}+\delta) = f(\bar{x}) + \sum_{i=1}^{\infty} \frac{1}{n!} \frac{\partial^n f(\bar{x})}{\partial x^n} \delta^n \tag{4.4}$$

where $\delta = x - \bar{x}$ is the independent variable perturbation around the nominal point. In this one-dimensional case, the partial derivative coincides with the ordinary derivative. When multi-variable functions are expanded, the partial derivative shall be used, hence the notation $\partial$ in Equation (4.4). The function $f(x)$ can represent either the objective or any constraint function indiscriminately. Hence, the first- and second-order finite-difference approximations introduced in the next sections hold for all of them.

### 4.1.1. FIRST-ORDER FINITE-DIFFERENCE

From Equation (4.4), the first-order derivative can be isolated as:

$$\frac{\partial f(\bar{x})}{\partial x} = \frac{f(\bar{x}+\delta) - f(\bar{x})}{\delta} - \sum_{i=2}^{\infty} \frac{1}{n!} \frac{\partial^n f(\bar{x})}{\partial x^n} \delta^{n-1} = \frac{f(\bar{x}+\delta) - f(\bar{x})}{\delta} + \mathcal{O}(\delta) \tag{4.5}$$

When the terms of order $\delta$ are truncated, the first-order derivative is approximated by a *forward-difference*:

$$\frac{\partial f(\bar{x})}{\partial x} \approx \frac{f(\bar{x}+\delta) - f(\bar{x})}{\delta} \tag{4.6}$$

with a truncation error of $\epsilon = \mathcal{O}(\delta) \approx \frac{\delta}{2} \frac{\partial^2 f(\bar{x})}{\partial x^2}$. To reduce the truncation error ($\delta << 0$, so higher powers are smaller), a *central-difference* scheme can be used. It expands for $f(x-\delta)$, subtracts

it by Equation (4.4), isolates the derivative and truncates the higher-order terms. The first-order derivative is then computed as:

$$\frac{\partial f(\bar{x})}{\partial x} \approx \frac{f(\bar{x}+\delta) - f(\bar{x}-\delta)}{2\delta} \tag{4.7}$$

with a truncation error of $\epsilon = \mathcal{O}(\delta^2) \approx \frac{\delta^2}{6}\frac{\partial^3 f(\bar{x})}{\partial x^3}$. As it employs two different Taylor expansions, it is also called second-order first-derivative approximation. The error can be further reduced by increasing the order of the approximation, i.e. employing Taylor expansions for $\bar{x} + -k\delta$. However, as WORHP uses second-order approximations, these schemes will not be explicitly explained.

When multi-variable functions are to be differentiate, i.e. the number of optimizable parameters $p$ is bigger than one, the same procedure applies for the derivative with respect to $p_j$ by defining the perturbation as $\boldsymbol{\delta}_j = [0, 0, \ldots, \delta, \ldots, 0]$, where the j-th element is the only non-zero one. Hence, to compute the objective gradient and the constraint Jacobian by second-order finite-difference approximations, the objective function and constraints shall be evaluated $2n_y$ times for dense matrices. In aerospace application, when often dynamical equations needs to be numerical integrated, this implicates $2n_y$ propagation of vector functions. WORHP employs group methods to reduce the number of function evaluations when the sparsity pattern of the derivative matrices is provided [26].

### 4.1.2. SECOND-ORDER FINITE-DIFFERENCE

The central finite-difference approximate of second derivatives is computed in a similar fashion as the central first-derivative, but the expansion for $f(x-\delta)$ is summed to Equation (4.4) to obtain:

$$\begin{aligned} f(\bar{x}+\delta) + f(\bar{x}-\delta) &= 2f(\bar{x}) + \sum_{i=1}^{\infty} \frac{1}{n!}\frac{\partial^n f(\bar{x})}{\partial x^n}\delta^n + \sum_{i=1}^{\infty} \frac{1}{n!}\frac{\partial^n f(\bar{x})}{\partial x^n}(-\delta)^n \\ f(\bar{x}+\delta) + f(\bar{x}-\delta) &= 2f(\bar{x}) + \frac{\partial^2 f(\bar{x})}{\partial x^2}\delta^2 + \mathcal{O}(\delta^4) \end{aligned} \tag{4.8}$$

which, after truncation of terms of order $\mathcal{O}(\delta^2)$, reduces to:

$$\frac{\partial^2 f(\bar{x})}{\partial x^2} = \frac{1}{\delta^2}\left[f(\bar{x}+\delta) - 2f(\bar{x}) + f(\bar{x}-\delta)\right] \tag{4.9}$$

Multi-variable functions require a multi-variable Taylor expansion. The computation of mixed derivative terms is not straightforwardly generalizable from the single-variable expression as in the case of first derivatives. A more detailed description can be found in [33]. When first derivatives are provided by the user, the Hessian matrix can be regarded as the Jacobian of the first-order information, simplifying the computation as outlined in Section 4.1.1.

### 4.1.3. BFGS METHOD

Computing the Hessian by second derivatives finite-difference approximation is extremely expensive when the number of optimizable parameters is medium-high. To overcome this computational burden, this information can be constructed by recursive updates (default option in WORHP). The new estimation of the Hessian is computed by a low-rank modification from the previous one. As this step shall preserve the symmetry and positive-definiteness properties, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) step is often used for unconstrained optimization problems:

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\Delta(\nabla J)[\Delta(\nabla J)]^T}{[\Delta(\nabla J)]^T \Delta\mathbf{y}} - \frac{\mathbf{H}_k \Delta\mathbf{y}(\Delta\mathbf{y})^T \mathbf{H}_k}{(\Delta\mathbf{y})^T \mathbf{H}_k \Delta\mathbf{y}} \tag{4.10}$$

where $\Delta(\nabla J) = \nabla J_{k+1} - \nabla J_k$ and $\Delta\mathbf{y} = \mathbf{y}_{k+1} - \mathbf{y}_k$, and the condition $[\Delta(\nabla J)]^T \Delta\mathbf{y} > 0$ is checked.

For constrained optimization, the recursive update term is much more complex. The background theory can be found in [24] [34], while an example of an algorithm for large-scale problems can be found in [35].

## 4.2. VARIATIONAL APPROACH

The major advantage of finite-difference approximation is that the functional relation between control parameters and the derivative of the dependent function shall not be explicitly known. However, for most applications in spaceflight, this relation can be analytically derived and exploited to reduce the computational burden associated with the derivative calculation. In the general case, the objective function $J$, the constraints $\mathbf{g}$, and consequently the Lagrangian $\mathscr{L}$, will be regarded as explicit functions of an independent variable $t$ (here assumed to be the commonly-used time variable, although there is no restriction on the choice), state $\mathbf{x}(t,\mathbf{u})$ and control $\mathbf{u}(t)$. This notation is illustrated in the following equations, but some implicit dependencies will be hidden in the remainder of the section.

$$\begin{aligned} &J(t_f,\mathbf{x}_f,\mathbf{u}_f),\mathbf{u}(t)) = \Phi(t_f,\mathbf{x}_f) + \int_{t_0}^{t_f} L(t,\mathbf{x}(t,\mathbf{u}),\mathbf{u}(t))dt \\ &\mathbf{g}_L \le \mathbf{g}(t,\mathbf{x}(t,\mathbf{u}),\mathbf{u}(t)) \le \mathbf{g}_U \\ &\mathscr{L}(t,\mathbf{x}(t,\mathbf{u}),\mathbf{u}(t)) = J(t,\mathbf{x}(t,\mathbf{u}),\mathbf{u}(t)) + \sum_{i=1}^{m_c} \mu_i g_i(t,\mathbf{x}(t,\mathbf{u}),\mathbf{u}(t)) \end{aligned} \tag{4.11}$$

where $g_L = g_U$ for equality constraints, and $t = \bar{t}$ for fixed-time constraints. When the lower and upper bounds are not constant, the constraint can be reformulated or split into two different ones in order to have the functional form in Equation (4.11)-2.

Applying the chain rule, a generic term $\partial f/\partial p_i$ can be expressed as:

$$\frac{\partial f}{\partial p_i}(t,\mathbf{x},\mathbf{u}) = \frac{\partial f}{\partial t}\frac{\partial t}{\partial p_i} + \sum_{k=1}^{n_s} \frac{\partial f}{\partial x_k}\frac{\partial x_k}{\partial p_i} + \sum_{k=1}^{n_c} \frac{\partial f}{\partial u_k}\frac{\partial u_k}{\partial p_i} \tag{4.12}$$

where $n_s$ is the number of state variables and $n_c$ the number of control variables. The functional form of the terms $\partial f/\partial t$, $\partial f/\partial x_k$ and $\partial f/\partial u_k$ are assumed to be known relations, otherwise the method is not applicable. The transition term $\partial x_k/\partial p_i$ is computed by numerical propagation of the variational equations that will be introduced in Section 4.2.1, while the term $\partial u_k/\partial p_i$ depends on the chosen control parameterization. This formulation can be directly applied to any constraints derivatives within the Jacobian in Equation (4.2).

On the contrary, following the objective functional form in Equation (4.11)-1, the terms $\partial J/\partial p_i$ of the objective gradient involve an integral term. Hence, the chain rule results in [2]:

$$\frac{\partial J(t)}{\partial p_i} = \frac{\partial \Phi(t_f)}{\partial p_i}\delta_{t,t_f} + \int_{t_0}^{t} \frac{\partial L}{\partial p_i}(\tau,\mathbf{x},\mathbf{u})d\tau + \frac{\partial t}{\partial p_i}L(t,\mathbf{x},\mathbf{u}) - \frac{\partial t_0}{\partial p_i}L(t_0,\mathbf{x}_0,\mathbf{u}_0) \tag{4.13}$$

where $\delta_{t,t_f}$ is the Kronecker delta of variables $t$ and $t_f$. The term $\partial L/\partial p_i(\tau,\mathbf{x},\mathbf{u})$ shall be further decomposed as:

$$\frac{\partial L}{\partial p_i}(\tau,\mathbf{x},\mathbf{u}) = \frac{\partial L}{\partial \tau}\frac{\partial \tau}{\partial p_i} + \sum_{k=1}^{n_s} \frac{\partial L}{\partial x_k}\frac{\partial x_k}{\partial p_i} + \sum_{k=1}^{n_c} \frac{\partial L}{\partial u_k}\frac{\partial u_k}{\partial p_i} \tag{4.14}$$

This formulation will involve the propagation of the *Lagrange* integral term in the objective function, as well as the propagation of the $n_y$ different $dL/dp_i$ terms as in Equation (4.14) for the first derivative (and ${n_y}^2$ terms for the second derivatives), increasing the computational load for each

iteration of the NLP solver. In the continuous problem, this is equivalent to increase the state vector dimension by adding one new variable defined as [3]:

$$\dot{x}_{n+1} = L(t,\mathbf{x}(t,\mathbf{u}),\mathbf{u}(t)), \quad x_{n+1}(t_0) = 0 \tag{4.15}$$

and redefine the objective function as:

$$J = x_{n+1}(t_f) \tag{4.16}$$

However, this one-variable dimension addition in the continuous problem results in a noteworthy increase of free parameters $n_y$, as well as additional defects constraints, in the transcribed problem.

An alternative consists in approximating the objective function integral by a quadrature formula:

$$J \approx \sum_{q=1}^{nodes} w_q L(t_q,\mathbf{x}_q,\mathbf{u}_q) \tag{4.17}$$

where $w_q$ are the weights associated to nodes $t_q$, which depend on the chosen quadrature scheme and number of nodes. The implemented Gauss-Legendre quadrature method and the formulae to compute the node-weight pairs will be discussed in Section 5.1.1. Hence, the derivatives can be applied directly to the quadrature approximation.

$$\frac{\partial J}{\partial p_i} \approx \sum_{q=1}^{nodes} w_q \left[ \frac{\partial L}{\partial t}\frac{\partial t}{\partial p_i} + \sum_{k=1}^{n_s} \frac{\partial L}{\partial x_k}\frac{\partial x_k}{\partial p_i} + \sum_{k=1}^{n_c} \frac{\partial L}{\partial u_k}\frac{\partial u_k}{\partial p_i} \right]_{(t_q,\mathbf{x}_q,\mathbf{u}_q)} \tag{4.18}$$

This approximation avoids both the increase in the associated NLP sub-problem and additional numerical propagations. In addition, in space trajectories applications the $\Delta V$ or propellant consumption profiles, usually used as objective functions, are smooth enough to use only a few quadrature nodes per sub-interval. For these reasons, the latter approach is implemented in the new-developed multiple-shooting tool. The introduced approximation error will be assessed empirically in Section 5.2.

Applying the chain rule again on Equation (4.12), the second derivatives can be computed as:

$$\begin{aligned}
\frac{\partial^2 f}{\partial p_j \partial p_i}(t,\mathbf{x},\mathbf{u}) &= \frac{\partial}{\partial p_j}\left\{ \frac{\partial f}{\partial t}\frac{\partial t}{\partial p_i} + \sum_{k=1}^{n_s} \frac{\partial f}{\partial x_k}\frac{\partial x_k}{\partial p_i} + \sum_{k=1}^{n_c} \frac{\partial f}{\partial u_k}\frac{\partial u_k}{\partial p_i} \right\} \\
&= \frac{\partial^2 f}{\partial p_j \partial t}\frac{\partial t}{\partial p_i} + \frac{\partial f}{\partial t}\frac{\partial^2 t}{\partial p_j \partial p_i} + \\
&+ \sum_{k=1}^{n_s} \left\{ \frac{\partial^2 f}{\partial p_j \partial x_k}\frac{\partial x_k}{\partial p_i} + \frac{\partial f}{\partial x_k}\frac{\partial^2 x_k}{\partial p_j \partial p_i} \right\} + \\
&+ \sum_{k=1}^{n_c} \left\{ \frac{\partial^2 f}{\partial p_j \partial u_k}\frac{\partial u_k}{\partial p_i} + \frac{\partial f}{\partial u_k}\frac{\partial^2 u_k}{\partial p_j \partial p_i} \right\}
\end{aligned} \tag{4.19}$$

Because the terms $\partial^2 f/\partial p_j \partial t$, $\partial^2 f/\partial p_j \partial x_k$ and $\partial^2 f/\partial p_j \partial u_k$ are mixed derivatives with respect to one variable and one parameter, they need to be further expanded by recursively application of the derivation chain rule.

$$\begin{aligned}
\frac{\partial^2 f}{\partial p_j \partial p_i}(t,\mathbf{x},\mathbf{u}) = &\left[ \frac{\partial^2 f}{\partial t^2}\frac{\partial t}{\partial p_j} + \sum_{h=1}^{n_s} \frac{\partial^2 f}{\partial x_h \partial t}\frac{\partial x_h}{\partial p_j} + \sum_{h=1}^{n_c} \frac{\partial^2 f}{\partial u_h \partial t}\frac{\partial u_h}{\partial p_j} \right]\frac{\partial t}{\partial p_i} + \frac{\partial f}{\partial t}\frac{\partial^2 t}{\partial p_j \partial p_i} + \\
&+ \sum_{k=1}^{n_s} \left[ \left( \frac{\partial^2 f}{\partial t \partial x_k}\frac{\partial t}{\partial p_j} + \sum_{h=1}^{n_s} \frac{\partial^2 f}{\partial x_h \partial x_k}\frac{\partial x_h}{\partial p_j} + \sum_{h=1}^{n_c} \frac{\partial^2 f}{\partial u_h \partial x_k}\frac{\partial u_h}{\partial p_j} \right)\frac{\partial x_k}{\partial p_i} + \frac{\partial f}{\partial x_k}\frac{\partial^2 x_k}{\partial p_j \partial p_i} \right] + \\
&+ \sum_{k=1}^{n_c} \left[ \left( \frac{\partial^2 f}{\partial t \partial u_k}\frac{\partial t}{\partial p_j} + \sum_{h=1}^{n_s} \frac{\partial^2 f}{\partial x_h \partial u_k}\frac{\partial x_h}{\partial p_j} + \sum_{h=1}^{n_c} \frac{\partial^2 f}{\partial u_h \partial u_k}\frac{\partial u_h}{\partial p_j} \right)\frac{\partial u_k}{\partial p_i} + \frac{\partial f}{\partial u_k}\frac{\partial^2 u_k}{\partial p_j \partial p_i} \right]
\end{aligned} \tag{4.20}$$

The terms directly involving a derivative of $f$ shall be known from its functional form, otherwise the method cannot be applied. The derivatives of $u$ are known from the chosen parameterization, while the derivatives of $x$ are computed by numerical propagation of the second-order variational equations to be presented in Section 4.2.2. Equation (4.20) holds for both the constraints and the quadrature approximation of the objective function. No acknowledgment of a second-order variational approach has been found in literature for optimization problems. Hence, Equations (4.19)-(4.20) as introduced in this Section, and the ones proposed in the following Section 4.2.2, are the outcome of an original analytical development by the thesis' author.

### 4.2.1. FIRST-ORDER VARIATIONAL EQUATIONS

To compute the first derivative information, we shall be able to differentiate the dependent function with respect to x at any time. Hence, the equations of motion for this first-order sensitivities can be set. By defining $S^k_{p_i} \triangleq \partial x_k / \partial p_i$, its time-derivative can be computed by differentiating the dynamical equations $\dot{x}_k = f_k(t, \mathbf{x}, \mathbf{u})$ by $p_i$ [2] [36] [37]:

$$\dot{S}^k_{p_i} = \frac{\partial f_k}{\partial t}\frac{\partial t}{\partial p_i} + \sum_{k=1}^{n_s} \frac{\partial f_k}{\partial x_k} S^k_{p_i} + \sum_{k=1}^{n_c} \frac{\partial f_k}{\partial u_k}\frac{\partial u_k}{\partial p_i} \tag{4.21}$$

These equations will now be specialized in detail for any parameter in a multiple-shooting transcription scheme, i.e. the initial state, the parameters of the control acceleration, and possibly the initial and final time in each sub-interval.

#### INITIAL STATE PARAMETER

In a generic sub-interval $[t_d, t_{d+1}]$ of the discretized problem, the sensitivity with respect to the initial state guess $x^g_i = x_i(t_d)$ will be denoted as $S^k_{x^g_i} \triangleq \partial x_k(t) / \partial x^g_i$. The different partial derivatives with respect to $x^g_i$ are:

$\frac{\partial t}{\partial x^g_i} = 0$ — A zero-term because the independent variable does not depend on the initial state guess.

$S^k_{x^g_i}(t) = \frac{\partial x_k(t)}{\partial x^g_i}$ — Transition term describing how the state at time $t$ would change due to a variation of the state initial guess. The initial condition is $S^k_{x^g_i}(t_d) = \delta_{k,i}$.

$\frac{\partial u_k}{\partial x^g_i} = 0$ — Zero-term because the control parameterization is independent of the initial state.

Hence, removing the zero terms in Equation (4.21) and reformulating in a matrix structure, the equations of motion for the state first sensitivities with respect to the state initial guess are:

$$\dot{\mathbf{S}}_x(t) = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial x^g_1} & \cdots & \frac{\partial \dot{x}_1}{\partial x^g_{n_s}} \\ \vdots & \vdots & \vdots \\ \frac{\partial \dot{x}_{n_s}}{\partial x^g_1} & \cdots & \frac{\partial \dot{x}_{n_s}}{\partial x^g_{n_s}} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{n_s}} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_{n_s}}{\partial x_1} & \cdots & \frac{\partial f_{n_s}}{\partial x_{n_s}} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial x^g_1} & \cdots & \frac{\partial x_1}{\partial x^g_{n_s}} \\ \vdots & \vdots & \vdots \\ \frac{\partial x_{n_s}}{\partial x^g_1} & \cdots & \frac{\partial x_{n_s}}{\partial x^g_{n_s}} \end{bmatrix} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{S}_x(t) \tag{4.22}$$

with initial condition $\mathbf{S}_x(t) = I$, a $(n_s \times n_s)$ identity matrix.

### CONTROL ACCELERATION PARAMETER

As will be explained in detail in Chapter 5, in the generic sub-interval $[t_d, t_{d+1}]$ the control profile $u_k(t)$ is discretized as a p-order polynomial:

$$u_k(t) = \sum_{j=0}^{n_p-1} a_j t^j \tag{4.23}$$

where $n_p$ is the number of control parameters per each of the $n_c$ control components $u_k(t)$. The optimizable parameters $u_k^p$ in the NLP sub-problem can be selected to be:

- The coefficients of the polynomial
- The control values at given nodes within the sub-interval, and the coefficients of the polynomial are computed by a Lagrange interpolation rule.

This choice deeply influences the computation of the partial derivative $\partial u_k/\partial u_k^p$, whose functional form is known. The partial derivative $\partial t/\partial u_k^p$ is again zero, while the sensitivity with respect to the control parameters $S^k_{u_k^p} \triangleq \partial x_k(t)/\partial u_k^p$ describes how the state would vary due to a variation in $u_k^p$. Equation (4.21) for the time-evolution of the first sensitivities with respect to these controllable parameters can be written in a matrix notation as:

$$\begin{aligned}
\dot{\mathbf{S}}_u(t) = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial u_1^1} & \cdots & \frac{\partial \dot{x}_1}{\partial u_{n_c}^{n_p}} \\ \vdots & \vdots & \vdots \\ \frac{\partial \dot{x}_{n_s}}{\partial u_1^1} & \cdots & \frac{\partial \dot{x}_{n_s}}{\partial u_{n_c}^{n_p}} \end{bmatrix} &= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{n_s}} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_{n_s}}{\partial x_1} & \cdots & \frac{\partial f_{n_s}}{\partial x_{n_s}} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial u_1^1} & \cdots & \frac{\partial x_1}{\partial u_{n_c}^{n_p}} \\ \vdots & \vdots & \vdots \\ \frac{\partial x_{n_s}}{\partial u_1^1} & \cdots & \frac{\partial x_{n_s}}{\partial u_{n_c}^{n_p}} \end{bmatrix} + \\
&+ \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_{n_c}} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_{n_s}}{\partial u_1} & \cdots & \frac{\partial f_{n_s}}{\partial u_{n_c}} \end{bmatrix} \begin{bmatrix} \frac{\partial u_1}{\partial u_1^1} & \cdots & \frac{\partial u_1}{\partial u_{n_c}^{n_p}} \\ \vdots & \vdots & \vdots \\ \frac{\partial u_{n_c}}{\partial u_1^1} & \cdots & \frac{\partial u_{n_c}}{\partial u_{n_c}^{n_p}} \end{bmatrix} \\
\dot{\mathbf{S}}_u(t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{S}_u(t) &+ \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{u}^p}
\end{aligned} \tag{4.24}$$

with initial condition $\mathbf{S}_u(t) = \mathbf{0}$, a $[n_s \times (n_c \cdot n_p)]$ zero matrix, and $\mathbf{u}^p$ is the vector containing the $n_c \cdot n_p$ control parameters.

### INITIAL TIME PARAMETER

To derive the equations of motion for $S^k_{t_d} = \partial x_k(t)/\partial t_d$, the sensitivity of the state with respect to the initial time of an interval, it is better to reformulate Equation (4.21) in the integral form [2]:

$$\frac{\partial x_k(t)}{\partial t_d} = \frac{\partial}{\partial t_d}\left[x_k(t_d) + \int_{t_d}^{t} f_k(\tau, \mathbf{x}, \mathbf{u}) d\tau\right] = \frac{\partial x_k(t_d)}{\partial t_d} + \int_{t_d}^{t} \frac{\partial \dot{x}_k(\tau)}{\partial t_d} d\tau + \frac{\partial t}{\partial t_d} f_k(t) - \frac{\partial t_d}{\partial t_d} f_k(t_d) \tag{4.25}$$

It is clear that the transition term is now:

$$\dot{\mathbf{S}}_{t_d}(t) = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial t_d} \\ \vdots \\ \frac{\partial \dot{x}_{n_s}}{\partial t_d} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{n_s}} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_{n_s}}{\partial x_1} & \cdots & \frac{\partial f_{n_s}}{\partial x_{n_s}} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial t_d} \\ \vdots \\ \frac{\partial x_{n_s}}{\partial t_d} \end{bmatrix} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \mathbf{S}_{t_d}(t) \tag{4.26}$$

with initial conditions $S^k_{t_d}(t) = (\partial x_k(t_d)/\partial t_d - f_k(t_d))$, for $k = 1, \dots, n_s$. The term $\partial x_k(t_d)/\partial t_d$ describes any explicit dependence of the initial state from initial time.

#### FINAL TIME PARAMETER

Rearranging Equation (4.25) for the final time derivative, the integral equation is:

$$\frac{\partial x_k(t)}{\partial t_{d+1}} = \frac{\partial}{\partial t_{d+1}} \left[ x_k(t_d) + \int_{t_d}^{t} f_k(\tau, \mathbf{x}, \mathbf{u}) d\tau \right] = \frac{\partial x_k(t_d)}{\partial t_{d+1}} + \int_{t_d}^{t} \frac{\partial \dot{x}_k(\tau)}{\partial t_{d+1}} d\tau + \frac{\partial t}{\partial t_{d+1}} f_k(t) - \frac{\partial t_d}{\partial t_{d+1}} f_k(t_d) \tag{4.27}$$

This derivative has no transition term since the initial condition is zero, and the term under the integral term stays zero strictly within the interval. At the final time, the derivative $\partial t/\partial t_{d+1}|_{t=t_{d+1}} = 1$, therefore:

$$S^k_{t_{d+1}} = \frac{\partial x_k(t)}{\partial t_{d+1}} = \begin{cases} f_k(t_{d+1}) & \text{, if } t = t_{d+1} \\ 0 & \text{, if } t \neq t_{d+1} \end{cases} \tag{4.28}$$

### 4.2.2. SECOND-ORDER VARIATIONAL EQUATIONS

As already outlined in Section 4.1.2, the Hessian matrix can be seen as the Jacobian of the first-order information. Therefore, to derive the equations of motion for the generic state second-order derivative $S^k_{p_j p_i} \triangleq \partial^2 x_k / \partial p_j \partial p_i$, Equation (4.21) shall be differentiated by $p_j$:

$$\begin{aligned} \dot{S}^k_{p_j p_i} &= \frac{\partial}{\partial p_j} \left[ \frac{\partial f_k}{\partial t} \frac{\partial t}{\partial p_i} + \sum_{k=1}^{n_s} \frac{\partial f_k}{\partial x_k} \frac{\partial x_k}{\partial p_i} + \sum_{k=1}^{n_c} \frac{\partial f_k}{\partial u_k} \frac{\partial u_k}{\partial p_i} \right] \\ &= \left[ \left( \frac{\partial^2 f_k}{\partial t^2} \frac{\partial t}{\partial p_j} + \sum_{h=1}^{n_s} \frac{\partial^2 f_k}{\partial x_h \partial t} S^h_{p_j} + \sum_{h=1}^{n_c} \frac{\partial^2 f_k}{\partial u_h \partial t} \frac{\partial u_h}{\partial p_j} \right) \frac{\partial t}{\partial p_i} + \frac{\partial f_k}{\partial t} \frac{\partial^2 t}{\partial p_j \partial p_i} \right] + \\ &+ \sum_{k=1}^{n_s} \left[ \left( \frac{\partial^2 f_k}{\partial t \partial x_k} \frac{\partial t}{\partial p_j} + \sum_{h=1}^{n_s} \frac{\partial^2 f_k}{\partial x_h \partial x_k} S^h_{p_j} + \sum_{h=1}^{n_c} \frac{\partial^2 f_k}{\partial u_h \partial x_k} \frac{\partial u_h}{\partial p_j} \right) S^k_{p_i} + \frac{\partial f_k}{\partial x_k} S^k_{p_j p_i} \right] + \\ &+ \sum_{k=1}^{n_c} \left[ \left( \frac{\partial^2 f_k}{\partial t \partial u_k} \frac{\partial t}{\partial p_j} + \sum_{h=1}^{n_s} \frac{\partial^2 f_k}{\partial x_h \partial u_k} S^h_{p_j} + \sum_{h=1}^{n_c} \frac{\partial^2 f_k}{\partial u_h \partial u_k} \frac{\partial u_h}{\partial p_j} \right) \frac{\partial u_k}{\partial p_i} + \frac{\partial f_k}{\partial u_k} \frac{\partial^2 u_k}{\partial p_j \partial p_i} \right] \end{aligned} \tag{4.29}$$

where a middle step, similar to Equation (4.19), has not been displayed. The term $\partial^2 t/\partial p_j \partial p_i$ is zero for any $p_j$ and $p_i$. The equations for the time-evolution of $x_k$ with respect to all the control parameters can be gathered in a matrix notation.

Assuming $f_k$ to be at least $C^2$*-continuous*, the Schwarz's theorem and the symmetric construction (Equation (4.29)) ensure the square $(1 + n_s + n_c \cdot n_p) \times (1 + n_s + n_c \cdot n_p)$ matrix $\dot{\mathbf{S}}_H$, defined in the following expression, to be symmetric.

$$\dot{\mathbf{S}}_H^k = \begin{bmatrix} \frac{\partial^2 \dot{x}_k}{\partial t_d^2} & \frac{\partial^2 \dot{x}_k}{\partial t_d \partial x_1^g} & \cdots & \frac{\partial^2 \dot{x}_k}{\partial t_d \partial x_{n_s}^g} & \frac{\partial^2 \dot{x}_k}{\partial t_d \partial u_1^1} & \cdots & \frac{\partial^2 \dot{x}_k}{\partial t_d \partial u_{n_c}^{n_p}} \\ \frac{\partial^2 \dot{x}_k}{\partial x_1^g \partial t_d} & \frac{\partial^2 \dot{x}_k}{\partial x_1^{g\,2}} & \cdots & \frac{\partial^2 \dot{x}_k}{\partial x_1^g \partial x_{n_s}^g} & \frac{\partial^2 \dot{x}_k}{\partial x_1^g \partial u_1^1} & \cdots & \frac{\partial^2 \dot{x}_k}{\partial x_1^g \partial u_{n_c}^{n_p}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ \frac{\partial^2 \dot{x}_k}{\partial u_{n_c}^{n_p} \partial t_d} & \frac{\partial^2 \dot{x}_k}{\partial u_{n_c}^{n_p} \partial x_1^g} & \cdots & \frac{\partial^2 \dot{x}_k}{\partial u_{n_c}^{n_p} \partial x_{n_s}^g} & \frac{\partial^2 \dot{x}_k}{\partial u_{n_c}^{n_p} \partial u_1^1} & \cdots & \frac{\partial^2 \dot{x}_k}{\partial u_{n_c}^{n_p\,2}} \end{bmatrix} \tag{4.30}$$

The first-order information can be gathered in a $(1 + n_s + n_c) \times (1 + n_s + n_c \cdot n_p)$ Jacobian matrix $\mathbf{S}_J$ :

$$\mathbf{S}_J = \begin{bmatrix} \delta_{t,t_d} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\partial x_1}{\partial t_d} & \frac{\partial x_1}{\partial x_1^g} & \cdots & \frac{\partial x_1}{\partial x_{n_s}^g} & \frac{\partial x_1}{\partial u_1^1} & \cdots & \frac{\partial x_1}{\partial u_{n_c}^{n_p}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial x_{n_s}}{\partial t_d} & \frac{\partial x_{n_s}}{\partial x_1^g} & \cdots & \frac{\partial x_{n_s}}{\partial x_{n_s}^g} & \frac{\partial x_{n_s}}{\partial u_1^1} & \cdots & \frac{\partial x_{n_s}}{\partial u_{n_c}^{n_p}} \\ 0 & 0 & \ldots & 0 & \frac{\partial u_1}{\partial u_1^1} & \cdots & \frac{\partial u_1}{\partial u_{n_c}^{n_p}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & \frac{\partial u_{n_c}}{\partial u_1^1} & \cdots & \frac{\partial u_{n_c}}{\partial u_{n_c}^{n_p}} \end{bmatrix} \tag{4.31}$$

where $\delta_{t,t_d}$ is the Kronecker delta for the variables $t$ and $t_d$. The bottom-left block has only zero elements because the control does not depend on the state or initial time. Given this notation, rearranging Equation (4.29) in a matrix form, the time-evolution of the second-order sensitivities can be written as:

$$\dot{\mathbf{S}}_H^k = \mathbf{S}_J^T \mathbf{H}_{f_k} \mathbf{S}_J + \sum_{h=1}^{n_s} \frac{\partial f_k}{\partial x_h} \mathbf{S}_H^h + \sum_{h=1}^{n_c} \frac{\partial f_k}{\partial u_h} \mathbf{H}_{u_k} \text{ , for } \quad k = 1, \ldots, n_s \tag{4.32}$$

with initial condition $\mathbf{S}_H^k = \mathbf{0}$, a zero matrix. In Equation (4.32), the matrix $\mathbf{H}_{f_k}$ is the $(1 + n_s + n_c) \times (1 + n_s + n_c)$ Hessian of $f_k$ with respect to the variables time, state and control, while $\mathbf{H}_{u_k}$ is the $(1 + n_s + n_c \cdot n_p) \times (1 + n_s + n_c \cdot n_p)$ Hessian of $u_k$ with respect to the control parameters. $\mathbf{H}_{u_k}$ is zero for a control defined as in Equation (4.23).

The second-order sensitivities involving the final time parameter can be derived by differentiating Equation (4.27) again. Clearly, as the sensitivity with respect to $t_{d+1}$ has no transition term, i.e. does not evolve with time, also this second derivative will be zero. Only at final time, when the derivative $\partial t / \partial t_{d+1}|_{t=t_{d+1}} = 1$, we could have a nonzero term:

$$S_{p_j t_{d+1}}^k = \frac{\partial^2 x_k(t)}{\partial p_j \partial t_{d+1}} = \begin{cases} \frac{\partial f_k(t_{d+1})}{\partial p_j} & \text{, if } t = t_{d+1} \\ 0 & \text{, if } t \neq t_{d+1} \end{cases} \tag{4.33}$$

### 4.2.3. TWO-BODY VARIATIONAL DYNAMICS

To illustrate an example of variational equations derivation, the classical not-thrusted two-body problem model, as presented in Section 2.2.1, will be analyzed. To have a clear and quick overview

of the dynamical system to differentiate, the equations of motion are repeated here:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = \begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ -\frac{\mu}{\|\mathbf{r}\|^3}\mathbf{r} \end{bmatrix} \tag{4.34}$$

Equation (4.22) shows how the partial derivatives of the dynamical equations with respect to the state are necessary ingredients to compute the objective gradient and the Jacobian of the constraints. The most complex term is the derivative of the velocity component with respect to a position variable, computed as follows:

$$\frac{\partial f_{v_k}}{\partial r_i} = \frac{\partial}{\partial r_i}\left[-\frac{\mu}{\|\mathbf{r}\|^3}r_k\right] = -\left(-\frac{3}{2}\frac{\mu}{\|\mathbf{r}\|^5}2r_i r_k\right) - \frac{\mu}{\|\mathbf{r}\|^3}\delta_{ik} = 3\frac{\mu}{\|\mathbf{r}\|^5}r_i r_k - \frac{\mu}{\|\mathbf{r}\|^3}\delta_{ik} \tag{4.35}$$

Given these relations, it is straightforward to derive the Jabobian of $\mathbf{f}(\mathbf{x})$ needed for the first-order variational equation:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \mu\frac{2r_x^2-r_y^2-r_z^2}{\|\mathbf{r}\|^5} & \mu\frac{3r_x r_y}{\|\mathbf{r}\|^5} & \mu\frac{3r_x r_z}{\|\mathbf{r}\|^5} & 0 & 0 & 0 \\ \mu\frac{3r_x r_y}{\|\mathbf{r}\|^5} & \mu\frac{2r_y^2-r_x^2-r_z^2}{\|\mathbf{r}\|^5} & \mu\frac{3r_y r_z}{\|\mathbf{r}\|^5} & 0 & 0 & 0 \\ \mu\frac{3r_x r_z}{\|\mathbf{r}\|^5} & \mu\frac{3r_y r_z}{\|\mathbf{r}\|^5} & \mu\frac{2r_z^2-r_x^2-r_y^2}{\|\mathbf{r}\|^5} & 0 & 0 & 0 \end{bmatrix} \tag{4.36}$$

Equation (4.32) describes the second-order variational equation in the general case. The first-order terms $\partial f_k/\partial x_h$ are easily derivable from the Jacobian in Equation (4.36), while the second-order terms require the Hessian $\mathbf{H}_{f_k}$ for each of the six scalar equations of motion. The Hessian of $f_k$ can be computed by partially differentiating the $k$-row of the Jacobian matrix with respect to all the variables. Hence, the second-order partial derivatives associated to $f_x$, $f_y$ and $f_z$, the dynamical equations of the position variables, are always zero since the first three rows of the Jacobian depend only on constant terms.

$$\mathbf{H}_{f_x} = \mathbf{H}_{f_y} = \mathbf{H}_{f_z} = \mathbf{0}_{6\times 6} \tag{4.37}$$

The Hessian of the velocity dynamical equations requires lengthy analytical manipulations. Therefore, symbolic software is a convenient alternative to compute them and simplify to the shortest notation. As an illustrative example, Maple 2016 calculates the functional form of $\mathbf{H}_{f_{v_x}}$ as:

$$\mathbf{H}_{f_{v_x}} = \frac{\mu}{\|\mathbf{r}\|^7} \begin{bmatrix} -3r_x\left(2r_x^2 - 3r_y^2 - 3r_z^2\right) & -3r_y\left(4r_x^2 - r_y^2 - r_z^2\right) & -3r_z\left(4r_x^2 - r_y^2 - r_z^2\right) & 0 & 0 & 0 \\ -3r_y\left(4r_x^2 - r_y^2 - r_z^2\right) & 3r_x\left(r_x^2 - 4r_y^2 + r_z^2\right) & -15r_x r_y r_z & 0 & 0 & 0 \\ -3r_z\left(4r_x^2 - r_y^2 - r_z^2\right) & -15r_x r_y r_z & 3r_x\left(r_x^2 + r_y^2 - 4r_z^2\right) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.38}$$

while similar expressions can be easily computed for $\mathbf{H}_{f_{v_y}}$ and $\mathbf{H}_{f_{v_z}}$.

# 5

# IMPLEMENTATION AND VALIDATION

An optimal control problem involves a continuous state and control profiles, $\mathbf{x}(t)$ and $\mathbf{u}(t)$. It is possible to consider it as the infinite-dimensional generalization of a non-linear programming problem, which involves a finite number of free control parameters $\mathbf{y}$ and constraints $\mathbf{g}$, as discussed in detail in Section 3.3. Since practical numerical methods can handle only a limited number of parameters, a transcription technique to convert the continuous problem into a finite-dimensional approximation is needed.

The trajectory optimization tool developed for this thesis research employs a multiple-shooting transcription scheme to discretize, and a hybrid NLP solver to optimize the free parameters. After each NLP step, the accuracy of the finite-dimensional approximation is assessed by optimality and feasibility checks, and the transcription and discrete optimization stages repeated if needed.

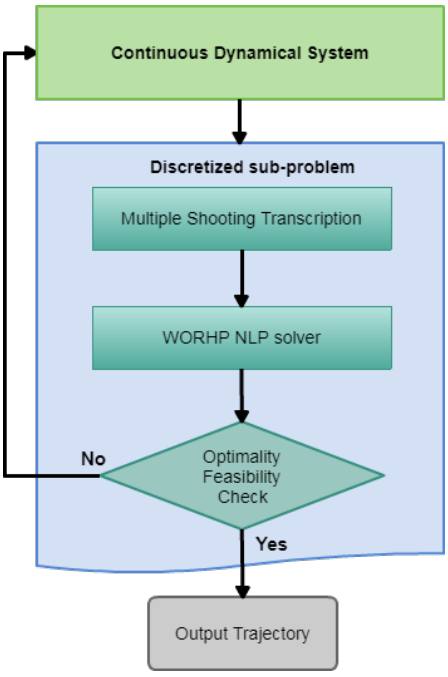



Figure 5.1: Flowchart of optimization tool

Several technical scenarios involve discontinuities in the state evolution. Examples in space applications can be impulsive maneuvers modelled as instantaneous velocity variations, lander mod-

ule droppings causing an immediate mass change, et cetera. In optimization, such problems are modelled as multi-phase, where a phase is defined as a trajectory segment in which the dynamical equations do not change and the state evolution is continuous. In the multiple-shooting framework, each phase is then divided into several sub-segments. In the remainder of the chapter, the discussion will focus on a single phase without any loss of generality. The multi-phase linkage conditions will be introduced in Section 5.1.5, while an exhaustive discussion can be found in [38].

Detailed principles for practical implementation of a multiple-shooting transcription scheme will be discussed in Section 5.1. The tool and its different features have been extensively tested and validated against a variety of test cases, and the process will be presented in Section 5.2.

## 5.1. MULTIPLE-SHOOTING IMPLEMENTATION

The multiple-shooting method requires an initial guess to start the transcription and the optimization process. The drawbacks of this method, such as the sensitivity of the final state to initial guess variations, lead to numerical problems when the initial guess is inaccurate. Furthermore, for practical applications the numerous numerical propagations and the method's local nature make it time-consuming when the initial guess is far from the optimum in the parameters' domain. For this reason, no random generation of the initial guess has been implemented, and it has to be provided by the user, both on the state and on the control.

While the state is numerically propagated from a given initial guess, the control evolution is parameterized selecting appropriate functional terms. A common and reasonable choice for a generic scalar $k$-component of the control is the univariate polynomial function as function of the independent variable $t$ (usually, but not necessarily, time):

$$u_k(t) = \sum_{j=0}^{n_p-1} a_j t^j \tag{5.1}$$

where $n_p$ is again the number of free parameters for each control component. As already outlined in Section 4.2.1, there are different possible choices for the free control parameters $u_k^p$, from which the coefficients $a_j$ are then computed. Two alternatives have been tested:

- The coefficients of the polynomial parameterization are the free parameters, therefore $u_k^{p_j} = a_j$ ;
- The free parameters are the control values at given nodes $t_i$ (see Section 5.1.1) within the sub-interval $u_k^{p_j} = u(t_j)$. Using the Lagrange interpolation formula, the coefficients can be computed imposing the following equality:

$$\sum_{j=0}^{n_p-1} a_j t^j = \sum_{i=1}^{n_p} \left[ u_k^{p_i} \prod_{\substack{w=1 \\ w\neq i}}^{n_p} \frac{t - t_w}{t_i - t_w} \right] \tag{5.2}$$

Although the first option is simple and does not require any intermediate passage, it is numerically problematic. Indeed, coefficients of different order usually have different orders of magnitude, and there is no straightforward way to select appropriate a priori scaling factors (see Section 5.1.4) for each of them. On the other hand, whilst the second option employs several middle passages and it is more difficult to differentiate, it requires a single scaling factor for all the free parameters, usually the maximum control magnitude.

### 5.1.1. PHASE DISCRETIZATION

The multiple-shooting transcription improves the stability of the overall optimization algorithms by introducing middle steps where the numerical integration is re-started. This approach helps to cut down the numerical round-off errors and the hypersensitivity to the initial guess. As already seen in Section 3.2.2, this requires the introduction of additional state guesses and defect constraints to ensure a continuous final trajectory.

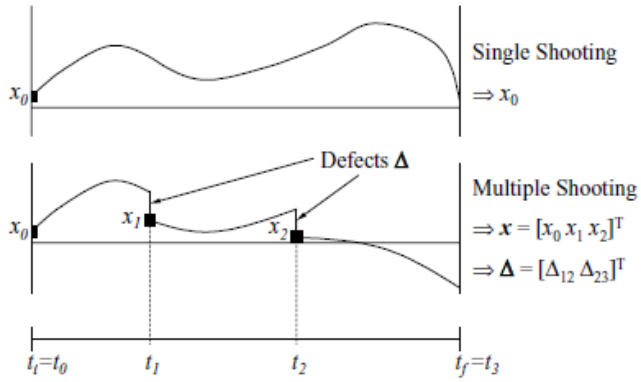



Figure 5.2: Single-state Multiple-Shooting discretization [21].

On top of this discretization, each interval is divided into several nodes:

- Control grid, which defines the nodes where the control value is a free parameter used as interpolating point;
- Path constraint grid, which defines nodes where to evaluate the path constraints as defined in Equation (4.11)-2;
- Quadrature grid, whose nodes are used to compute the objective function by quadrature approximation, as explained in Section 4.2.

The union of these nodes results in the final integration grid, where the state and sensitivity values are needed for the NLP step.

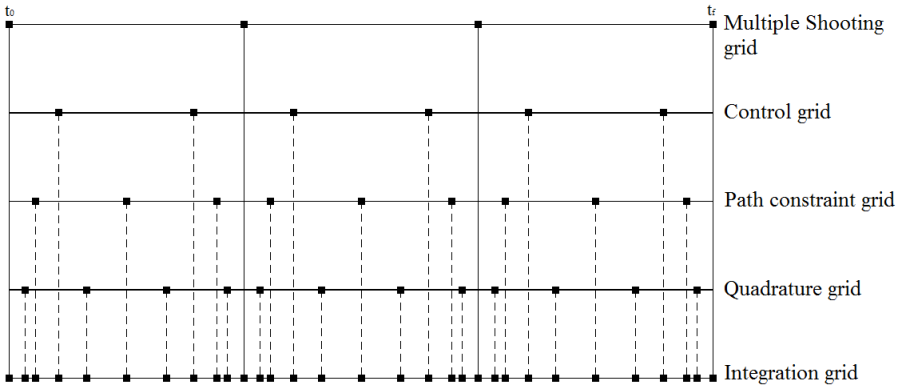



Figure 5.3: Different components of grid discretization

To compute the nodes' position, and the associated weights for the control, path constraints and quadrature grids, the Gauss-Legendre quadrature has been selected for its accuracy. Indeed, it

yields exact results for polynomials of order $2n+1$ with only $n$ nodes. The $n$ nodes are the roots of the Legendre polynomial of degree $n$, given by the relation:

$$P_n(x) = 2^n \sum_{k=0}^{n} x^k \binom{n}{k} \binom{\frac{n+k-1}{2}}{n} \tag{5.3}$$

while the associated weights can be computed as: $w_i = \frac{2}{(1-x_i^2)P_n'(x_i)^2}$.

### 5.1.2. SPARSE APPROACH FOR MULTIPLE-SHOOTING

Discretizing the trajectory causes non-subsequent segments to be directly independent, which mathematically translates into sparse Jacobian and Hessian matrices. In detail, the multiple-shooting transcription discretizes the independent variable interval into $(m-1)$ sub-segments. Consequently, $n_s(m-1)$ defect constraints are introduced to ensure continuity on the state variables evolution, each of which depends only on the controllable parameters of a specific sub-interval and its successive neighbor. Specifically, only $[n_s(n_s+n_c\cdot n_p)(m-1)+n_s(m-2)]$ elements of the Jacobian are nonzero out of $[n_s(n_s+n_c\cdot n_p)(m-1)^2]$ elements in the matrix. Therefore, the percentage of nonzero elements grows approximately as $[1/(m-1)]$ as the number of discretization intervals increase, and the enhanced sparsity computationally counteracts the increased number of variables for the associated discretized problem. In particular, the ordering of free parameters and constraints is essential to construct block-diagonal Jacobian and Hessian matrices, a pattern that highly speeds up the NLP iteration.

For example, gathering the variables and constraints by segments, the sparsity pattern of the Jacobian of a problem with $n_s = 4$ state variables, and a single piece-wise constant control variable $n_c \cdot n_p = 1$, discretized in 10 sub-intervals, is depicted in Figure 5.4, resulting in 11.8% nonzero elements, where there are as many rows as constraints and columns as free parameters.

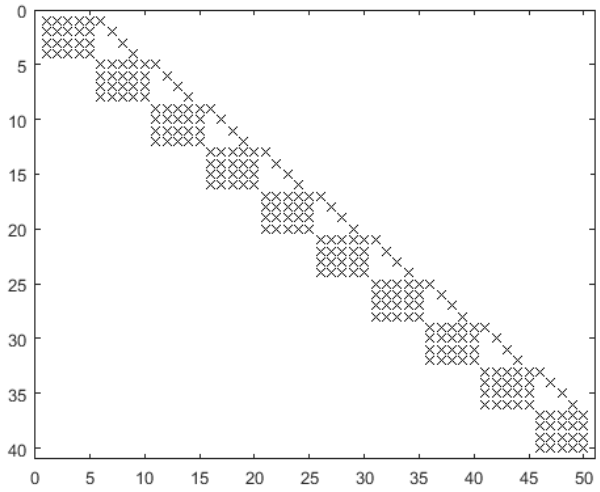



Figure 5.4: Sparsity pattern of defect constraints' Jacobian.

For what concerns the second-order information, the various segments are completely independent. Hence, only $[(n_s+n_c\cdot n_p)^2(m-1)]$ elements of the Hessian are nonzero out of $[(n_s+n_c\cdot n_p)(m-1)]^2$, leading to a sparsity percentage exactly equal to $[1/(m-1)]$. For the example above, this percentage is 10.0%, and the sparsity pattern is illustrated in Figure 5.5, where the number of rows and columns coincides with the number of free parameters.

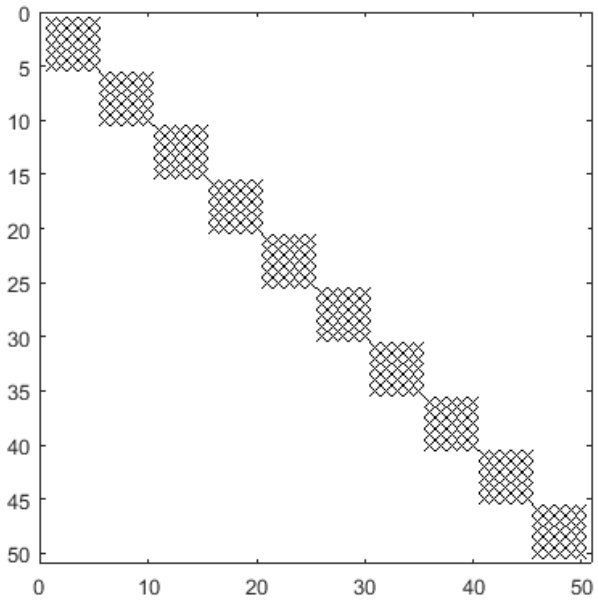



Figure 5.5: Sparsity pattern of the augmented Lagrangian's Hessian.

### 5.1.3. WORHP NLP SOLVER

Among the different interfaces offered by WORHP [26], the C++ library *Full Feature Interface* has been used. It is the most flexible and powerful interface to the NLP solver as it employs the *Reverse communication* paradigm, in contrast to the most simple and common *Direct communication*. With the latter, the NLP solver requires pointers to functions computing the objective index and constraint's violations, and then it computes the required information within the solver's call. On the contrary, in the reverse approach, the NLP solver is inserted in an outer loop where it performs an NLP step and it instructs the user to update the required information for the successive iteration. The iteration loop that usually runs inside a *Direct Commnication* solver has to be provided by the caller. In short, the Full Interface requires the user to perform an action instead of doing it itself. In this way, the caller directly modifies the values in the corresponding C++ structs. This choice allows extreme flexibility, in particular when the sparsity pattern of the derivative matrices and their nonzero values shall be provided.

Indeed, due to its fundamental suitability for large-scale problems, WORHP works properly on sparse matrices, avoiding zero entries for memory and computational performance. The sparsity pattern is characteristic of a problem and constant at each iteration, while the actual nonzero values depend on the current point $\mathbf{x}$. WORHP internally employs the *Coordinate storage* format to represent sparse matrices, denoting a nonzero element by the triplets $a_k$=($val, row, col$) for $k = 1, \dots, n_{nz}$. These triplets shall be provided in a column-wise order. For what concerns the Hessian, only the lower triangular part is saved, still following the column-major order for the strictly lower part, followed by the diagonal elements. Whilst these rules improve the algorithm efficiency and compatibility with Fortran 95 structs, WORHP's core language, they make the transcription-NLP solver interface only specialized for this tool. Other solvers can be integrated within the variational multiple shooting, but other interfaces shall be developed and a flag system designed to select the proper one. While a comparison of different NLP solvers' performance could be interesting for practical problems, it is beyond the purpose of this thesis, aimed to analyze the variational approach on the transcription side of the whole optimization tool.

### 5.1.4. PARAMETER AND FUNCTION SCALING

For any numerical algorithm, scaling is a critical aspect. In optimization, it influences numerical conditioning, convergence rate, termination tests, et cetera [2]. While a detailed discussion on scaling can be found in [24], some general rules to have a well-conditioned associated NLP subproblem are [2]:

- Free variables defined in the same range, usually $-1 \leq x \leq 1$;
- Dependent functions of the same order of magnitude, usually $J \approx g_1 \approx \cdots \approx g_m \approx 1$;
- Scale the dependent functions such that the Lagrange multipliers are close to one;
- The rows and columns of the Jacobian to be of the same order of magnitude;
- Scale the quantities such that the condition number of the Hessian is close to one;
- Scale the problem such that the condition number of the KKT matrix (Equation (3.21)) is close to one.

These hints are often conflicting and not implementable in an automatic procedure. For example, in a generic case it is not possible to simultaneously scale both the magnitude of the constraints and the rows of the Jacobian, i.e. the constraints' derivatives. WORHP already implements the rule on the KKT condition number, and experimentally the scaling of the objective function and constraints based on their gradients.

On top of that, in this novel multiple-shooting transcription tool the free variables and the dependent functions are scaled accordingly to an associated characteristic magnitude optionally supplied by the user. Consequently, the same quantity is used to scale the associated derivatives. Two reasons have driven this implementation choice: variables and functions in the same range help the optimizer convergence rate, as suggested from the hints outlined above; the user is able to manually scale a particular problem based on its specific characteristics. For example, the orbital elements' range for a specific trajectory is well known, with angular quantities spanning from 0 to $\pi$ (inclination), from 0 to $2\pi$ (argument of pericenter, etc.), and length quantities spanning from 0 to $10^9 - 10^{10}$ km for interplanetary trajectories. Once having defined a scaling factor $s_j$, the new scaled parameter $\widetilde{p}_j$ is defined by a linear transformation as:

$$\widetilde{p}_j = s_j p_j \tag{5.4}$$

#### PARAMETER SCALING

From continuous state and control variables, the multiple shooting generates a multiple number of associated free parameters to optimize, corresponding to the initial guess and control parameters at each sub-interval. The same variable can assume a wide range of values along the time span, and therefore different discretized parameters associated with the same continuous variable can be of different orders of magnitude. Hence, the ideal situation would be to associate a different scale to each discretized variable. All these scales cannot be supplied manually by the user, therefore an automatic procedure would be necessary. A possibility is to define the scale as:

$$s_j = \frac{1}{|p_j^g|} \tag{5.5}$$

where $p_j^g$ is the first initial guess of the parameter. However, this choice leads two major drawbacks:

- the user cannot directly supply a scale;

- the parameters can highly deviate from the supplied initial guess and the scale be no longer suitable.

For this reason, the alternative of a single scaling factor (usually its maximum magnitude) for each continuous variable has been implemented. This choice preserves the user interface flexibility, allowing to define each discretized parameter within unit bounds. Assuming that the scaling is the same over all the sub-intervals, we can group the free parameters in the following matrix form:

$$\begin{bmatrix} \widetilde{\mathbf{x}} \\ \widetilde{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_u \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \tag{5.6}$$

where $\mathbf{W}_x$ and $\mathbf{W}_u$ are diagonal scaling matrices for the state and control parameters.

### CONSTRAINT SCALING

For what concerns the constraints, naming $\mathbf{g}_d = \mathbf{0}$ the defects and $\mathbf{g}_p = \mathbf{0}$ the discretized path constraints, their scaled form is:

$$\widetilde{g} = \begin{bmatrix} \mathbf{W}_{g_d} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{g_p} \end{bmatrix} \begin{bmatrix} \mathbf{g}_d \\ \mathbf{g}_p \end{bmatrix} \tag{5.7}$$

where $\mathbf{W}_{g_d}$ and $\mathbf{W}_{g_p}$ are diagonal scaling matrices. Hence, the scaled Jacobian matrix results to be:

$$\widetilde{G} = \begin{bmatrix} \mathbf{W}_{g_d} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{g_p} \end{bmatrix} G \begin{bmatrix} \mathbf{W}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_u \end{bmatrix}^{-1} \tag{5.8}$$

As shortly outlined in [2], the defect constraint can be written in the symbolic form:

$$\mathbf{g}_d = \mathbf{x} - \int_{t_i}^{t_{i+1}} \mathbf{f}\, dt \tag{5.9}$$

The unscaled Jacobian follows as:

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} - \int_{t_i}^{t_{i+1}} \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\, dt & -\int_{t_i}^{t_{i+1}} \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\, dt \\ \frac{\partial \mathbf{g_p}}{\partial \mathbf{x}} & \frac{\partial \mathbf{g_p}}{\partial \mathbf{u}} \end{bmatrix} \tag{5.10}$$

As the time step becomes smaller, i.e. $(t_{i+1} - t_i) \to 0$, the Jacobian block corresponding to the continuity conditions tends to $\mathbf{G}_{g_d} \sim \mathbf{I}$, improving the conditioning of the original Jacobian. Hence, to preserve this property, the defect constraint is scaled using the same scale of the corresponding state variable:

$$\mathbf{W}_{g_d} = \mathbf{W}_x \tag{5.11}$$

For the path constraints, the user shall supply a suitable scale in order to well-scale their corresponding rows, because no a priori information can be exploited in the general case.

### OBJECTIVE FUNCTION SCALING

The objective function is scaled using a user-supplied characteristic value, such that its magnitude is around one. WORHP has another scaling procedure for the objective function as [26]:

$$\widetilde{J} = \begin{cases} J, & \text{if } J < \text{ScaleFacObj} \\ \text{ScaleFacObj}, & \text{otherwise} \end{cases} \tag{5.12}$$

where ScaleFacObj is 10 by default. This further justifies why it is convenient to scale $J$ around one, ensuring its value to remain below that threshold.

In addition, the optimality criterion, which usually requires a simple condition on the infinity norm of the first derivative of the augmented objective as

$$\|\nabla_x L\|_\infty \leq \text{TolOpti}, \tag{5.13}$$

is substituted by a scaled version as

$$\|\nabla_x L\|_\infty \leq \frac{\text{TolOpti} \cdot \max(1, |J|) + \|\boldsymbol{\mu}^T \mathbf{g}\|_\infty}{\|\mathbf{d}\|_\infty} \tag{5.14}$$

where $\boldsymbol{\mu}$ are the Lagrange multipliers associated to the constraints $\mathbf{g}$, and $\mathbf{d}$ is the search direction computed by the quadratic subproblem at the current point. This scaled version relaxes the condition in Equation (5.13), which can result too strict for numerical purposes and sometimes impossible to satisfy with computer arithmetic.

#### INDEPENDENT VARIABLE SCALING

As depicted in Figure 5.3, the full interval is subdivided into an integration grid, as a result of different discretizations. In order to keep the integration grid constant when the initial or final times (or any other independent variable) are free, the phase is normalized between zero and one with respect to a scaled variable defined as:

$$\tau = \frac{t - t_0}{t_f - t_0} \tag{5.15}$$

This independent-variable change requires a modification in the dynamical equations as:

$$\begin{aligned} \frac{d\mathbf{x}}{d\tau} &= \frac{d\mathbf{x}}{dt} \cdot \frac{dt}{d\tau} = \mathbf{f}(t, \mathbf{x}, \mathbf{u}) \cdot (t_f - t_0) \\ \int_{t_0}^{t_f} L dt &= \int_0^1 L \frac{dt}{d\tau} d\tau = \int_0^1 L \cdot (t_f - t_0) d\tau \end{aligned} \tag{5.16}$$

In particular, this expedient is useful when the gradients with respect to a time parameter are to be computed by finite-difference approximations, avoiding the time-perturbed trajectories to change the integration grid several times and degrade the derivative accuracy.

### 5.1.5. MULTI-PHASE

The entire discussion presented in this chapter can be generalized for multi-phase problems. A generic multi-phase problem is set as a collection of sequential single phases. Different phases can have different discretization settings, scales and control parameterizations. The phases are then connected by linking conditions which ensure time-continuity and, optionally, some state variables continuity. For example, space trajectories with impulsive maneuvers are modelled as multi-phase problems with position-continuity among the different phases, but not velocity-continuity conditions (see Section 6.2). In Figure 5.6, the Jacobian matrix of a multi-phase problem is depicted. In detail, the number of state variables $n_s = 4$, and the control parameters $n_c \cdot n_p = 2$, while the first and third phase has been discretized in three sub-intervals, whereas the second in two segments. The linking conditions ensure the continuity of the first two state variables, while the other two can be discontinuous.

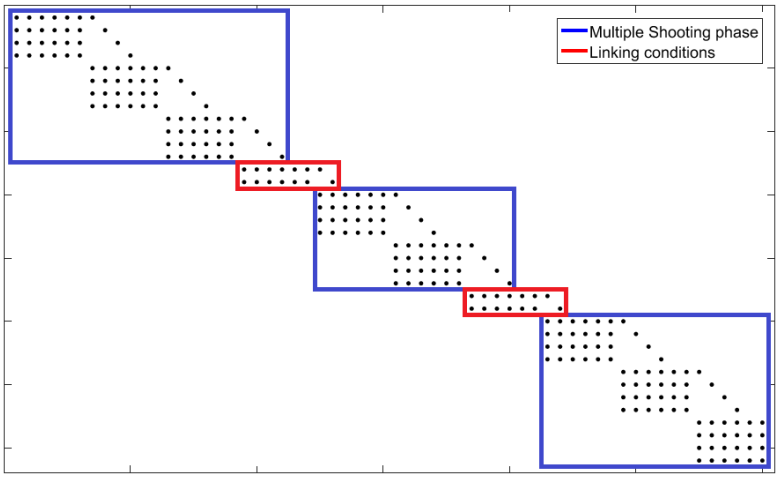



Figure 5.6: Sparsity pattern of Jacobian matrix for a multi-phase problem.

### 5.1.6. TOOL CAPABILITIES

In the design and development process of the *variational Multiple Shooting*, the tool flexibility has been one key aspect to ensure its applicability to the majority of continuous optimization problems. To this end, the control acceleration has been set as optional, allowing the tool to be suitable for:

- Optimal control problems, with or without coasting arcs, e.g. low-thrust trajectories (see Section 6.1);
- Multi-phase boundary-value-problems (BVP) with an objective index, e.g. impulsive maneuver trajectories (see Section 6.2);
- Initial-value-problems (IVP) or boundary-value-problems (BVP) without objective function, e.g. cannon-ball problem.

In the latter case, the automatic "trick" is to define a trivial objective function and switch WORHP on *feasibility-only* mode.

Different flags allow to singularly disable the variational dynamics on the gradient, Jacobian or Hessian. In this way, the variational approach on different components can be assessed, as well as the finite-difference approximation can be used to compute derivative information when the partial derivatives are not derivable for a specific quantity. Settable values allow the user to define optimal parameters for the problem at hand, such as the number of shooting sub-intervals, control nodes and control parameterization, path constraints nodes, integration steps, et cetera.

In addition to that, a plug-and-play approach has been employed to interface different blocks with the transcription scheme. For example, without digging into the implementation details, this approach allows the user to choose any numerical integrator (at least any available in the ACE software repository SMART [39]), without limiting the choice to only a few embedded schemes. This advantageous characteristic is useful when a specific problem shall be solved. For example, adaptive-stepsize integrators are not suitable when finite-different approximations are enabled because the stepsize sequence of the reference trajectory integration would probably be different from the one of the different perturbed trajectories, yielding a degradation of the derivative computation. On the contrary, when the variational dynamics is enabled, these more efficient and sophisticated schemes can be applied as all the quantities are propagated at once (sort of *internal differentiation* [40]), resulting in a unique stepsize sequence. Detailed information on integration techniques suitable for transcription schemes can be found in [41], [42] and [43].

## 5.2. SOFTWARE VERIFICATION AND VALIDATION

All the theoretical and practical information discussed in the current and previous chapters led to the development of a C++ software within the SMART repository [39]. Before the actual application to practical cases, the tool needed extensive verification to ensure it to be error-free and well-engineered. Indeed, several code refactoring and architecture rearrangements were driven by considerations which emerged during the test phase. This section will introduce different test cases with an analytical or well-known solution, useful to assess the software reliability and performance. In particular, the latter is fundamental from a validation perspective as we shall ensure that the built tool is actually more efficient than the typical finite-difference counterpart. To analyze these characteristics, in this testing phase it is not crucial how big, stiff or complex the example is, but it is important to know the optimal solution and to test different features. On the contrary, the next chapter will introduce large-scale and complex optimization problems, and multiple-shooting will be used as a completed algorithm, whose performance is to be further assessed.

With these goals in mind, the verification will be performed in two steps:

- Check the derivative computation with WORHP's embedded feature for derivative verification. Figure 5.7 shows the output of the derivative check, in specific the Hessian nonzero elements. The second and third columns report the user-supplied derivatives and the finite-difference values respectively. Two threshold are used to check the discrepancy, an absolute and a relative one, both set to $10^{-3}$ by default. When the former is not respected, as checked by the second-to-last column, the row is yellow coloured. When the relative error is above the correspondent threshold, as checked by the last column, the row becomes red;
- Apply the tool to the test case and check similarity of the solution with the known one. When only plots of the solution are available, a qualitative analysis shall be carried out.

```
  HM(12,10) = DL/Dx12x10    3.4201280562E-01  3.4195610773E-01  5.67E-05  5.67E-05
  HM(12,11) = DL/Dx12x11    3.1842266827E+00  3.1841785374E+00  4.81E-05  1.51E-05
* HM(14,13) = DL/Dx14x13    2.4216914582E+01  2.4218255332E+01  1.34E-03  5.54E-05
  HM(15,13) = DL/Dx15x13    2.8960800874E+00  2.8962754052E+00  1.95E-04  6.74E-05
  HM(16,13) = DL/Dx16x13    2.3525996384E-01  2.3524241676E-01  1.75E-05  1.75E-05
  HM(15,14) = DL/Dx15x14    4.3242382393E+00  4.3242113692E+00  2.69E-05  6.21E-06
  HM(16,14) = DL/Dx16x14    3.5957302583E-01  3.5954090493E-01  3.21E-05  3.21E-05
  HM(16,15) = DL/Dx16x15    3.1862602861E+00  3.1862351308E+00  2.52E-05  7.89E-06
* HM(18,17) = DL/Dx18x17    2.1898658826E+01  2.1900488832E+01  1.83E-03  8.36E-05
  HM(19,17) = DL/Dx19x17    2.5044390052E+00  2.5046817223E+00  2.43E-04  9.69E-05
  HM(20,17) = DL/Dx20x17    1.9863364003E-01  1.9862572587E-01  7.91E-06  7.91E-06
  HM(19,18) = DL/Dx19x18    4.3829606075E+00  4.3829659865E+00  5.38E-06  1.23E-06
  HM(20,18) = DL/Dx20x18    3.6368667763E-01  3.6367293788E-01  1.37E-05  1.37E-05
  HM(20,19) = DL/Dx20x19    3.1866512366E+00  3.1866353573E+00  1.59E-05  4.98E-06
* HM(22,21) = DL/Dx22x21    1.7191887031E+01  1.7193931825E+01  2.04E-03  1.19E-04
  HM(23,21) = DL/Dx23x21    1.8611230769E+00  1.8613765831E+00  2.54E-04  1.36E-04
  HM(24,21) = DL/Dx24x21    1.4297039204E-01  1.4299063593E-01  2.02E-05  2.02E-05
```

Figure 5.7: Illustration of WORHP derivative check tool interface.

Along with the verification phase, the validation will be carried out by switching on and off the variational dynamics and comparing the results and CPU-time performance of these alternatives. When possible and meaningful, the variational approach performance will be compared with the ones in literature as well. On the other hand, the number of function and constraint evaluations would not be meaningful indicators as the finite-difference approximations call them numerous times by construction (see Section 4.1) each iteration, while the variational approach calls them only once.

### 5.2.1. VAN DER POL PROBLEM

The Van Der Pol example is the first test case solved with a multiple-shooting algorithm, as reported in the paper by H. Bock and K. Plitt [44]:

$$
\begin{aligned}
&min \;\; J = \frac{1}{2}\int_0^5 (x_1^2 + x_2^2 + u^2)\,dt \\
&s.t.: \dot{x}_1 = x_2, && x_1(0) = 1 \\
&\quad \dot{x}_2 = -x_1 + (1 - x_1^2)x_2 + u\,, && x_2(0) = 0 \\
&\quad x_1(5) - x_2(5) + 1 = 0
\end{aligned}
\tag{5.17}
$$

This relatively straightforward two-state optimal control problem is suitable for tool testing, validation and performance assessment. For comparison purposes, the same settings as by [44] are employed. Hence, the time interval is discretized into $m = 3, 6, 11, 21$ mesh points, and the control is parameterized by a piecewise linear polynomial. The initial conditions are chosen as $x_1(0) = 1$, $x_2(0) = 0$ and $u(t) = 0.5$ for $0 \leq t \leq 5$, following the specification in [44]. The results can be assessed only by a qualitative point of view, comparing the original plots with the reproduced ones in Figure 5.8. The results are qualitatively well reproduced, and the final constraint is respected to $10^{-9}$, below the set threshold.

The derivative computation has been checked against the numerical ones, computed by finite differences, thanks to the tool discussed in Section 5.2, and the results are shown in Table 5.1.

| Derivative type | # Checked derivatives | Differing absolutely | Differing relatively | Differing both |
|---|---|---|---|---|
| DF | 80 | 0 | 0 | 0 |
| DG | 202 | 0 | 0 | 0 |
| HM | 200 | 5 | 0 | 0 |

Table 5.1: Van Der Pol derivative check.

The few second-order derivatives which differ absolutely are below the relative threshold, therefore the derivative check process is passed. Given the outcomes as above, the verification on this example can be judged as positively terminated.

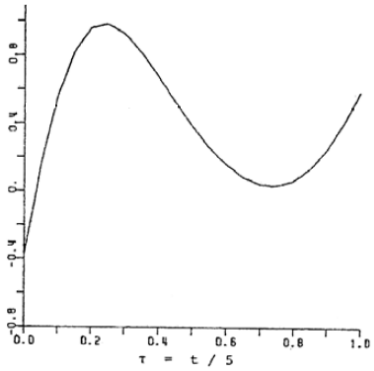
For what concerns the validation and performance assessment, we shall show that the variational approach actually improves the computational performance. [44] shows the evolution of the objective index as a function of the number of discretization intervals. The results are reproduced and compared in the table as follows.

| Mesh | J[44] | J | CPU [s][44] | CPU [s] |
|---|---|---|---|---|
| 3 | 1.8360 | 1.8360 | 0.174 | 0.019 |
| 6 | 1.6907 | 1.6869 | 0.241 | 0.022 |
| 11 | 1.6860 | 1.6859 | 0.305 | 0.041 |
| 21 | 1.6857 | 1.6857 | 0.813 | 0.064 |

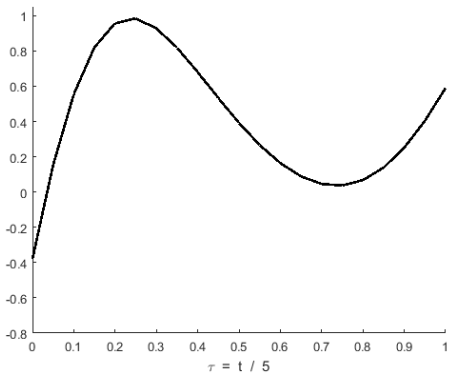
Table 5.2: Van Der Pol result comparison.

Table 5.2 shows an improvement of a factor 10 with equal or better results for the same number of mesh points. However, different NLP solvers and different computational resources have been used to point out the variational approach as the sole cause of this improvement. To better analyze the effect of the variational dynamics, Table 5.3 shows the relative computational times (rounded average after several simulations), using the same computational desktop, when it is applied on different derivative information.

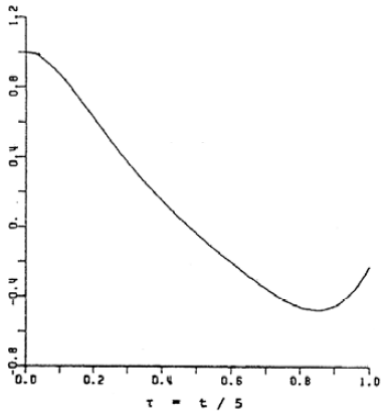
As the relative computational time is always bigger than one in the third column, the full variational approach (in line one DF-DG-HM notation points out that all the derivative quantities have been



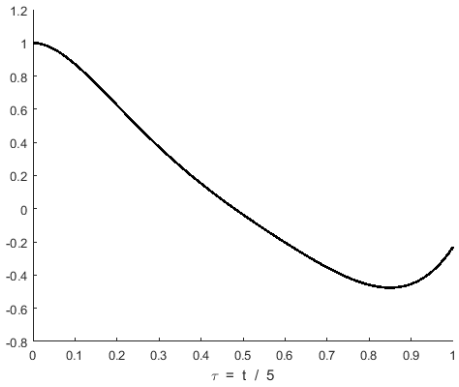

(a) VDP control profile from [44].



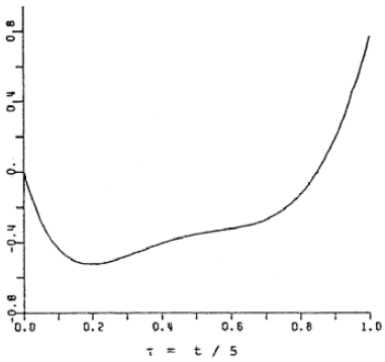

(b) VDP control profile, variational multiple-shooting.



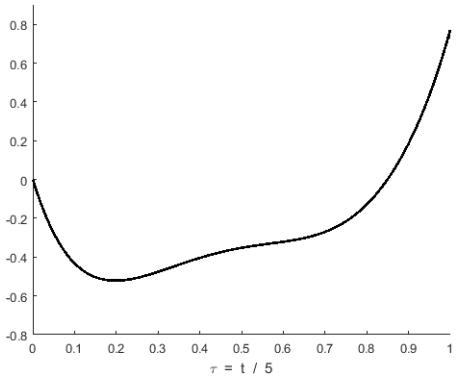

(c) VDP state $x_1$ profile from [44].




(d) VDP state $x_1$ profile, variational multiple-shooting.




(e) VDP state $x_2$ profile from [44].




(f) VDP state $x_2$ profile, variational multiple-shooting.

Figure 5.8: Van Der Pol example resulting profiles' comparison.

| Variational Derivative | J | Rel. CPU [-] | # NLP Iterations |
|---|---|---|---|
| DF-DG-HM | 1.6857 | - | 7 |
| (a)DF-DG | 1.6857 | 1.5 | 28 |
| (b)DF-DG | 1.6857 | 6.0 | 7 |
| DF-HM | 1.6857 | 2.0 | 7 |
| DG-HM | 1.6857 | 3.0 | 7 |
| DF | 1.6857 | 3.0 | 28 |
| DG | 1.6857 | 4.0 | 28 |
| HM | 1.6857 | 4.5 | 7 |
| none | 1.6857 | 8.0 | 28 |

Table 5.3: Van Der Pol variational performance comparison. For explanation see text.

computed by variational approach) results to be the most efficient one. In line two, the second-order derivatives for the Hessian HM are computed using the BFGS method, as explained in Section 4.1.3. For this reason, the number of iterations is higher, while the performance does not worsen excessively as the first-order information used for the recursive updates is still computed using the variational dynamics. On the contrary, row three uses the second-order finite-differences, as shown in Section 4.1.2, and the performance gets considerably worse as 200 derivatives (see Table 5.1) shall be computed by perturbation propagations. The same reasoning holds when only the first-order derivatives DF-DG are computed by finite approximation (row eight), but with better performance as the first-order finite differences are faster to compute. The single contribution of the first-order information is investigated by switching off the variational approach individually on the objective's gradient DF, as in rows five and seven, or the constraints' Jacobian DG, as in rows four and six. Even though the number of nonzero elements of DG is more than double the ones in DF (see Table 5.1), we can see that the variational dynamics is more influential on DG than on DF. This results from the application of group methods, as explained in Section 4.1.1, to reduce the number of perturbations to be propagated for the Jacobian matrix, methods that cannot be applied for the gradient vector. Finally, the last line shows the performance deterioration when the full finite difference approach is used. The objective function value has been reported for each variational-finite difference combination to verify that the optimization converges to the same minimum and that this element does not corrupt the previous analysis.

### 5.2.2. 1D TEST CASE

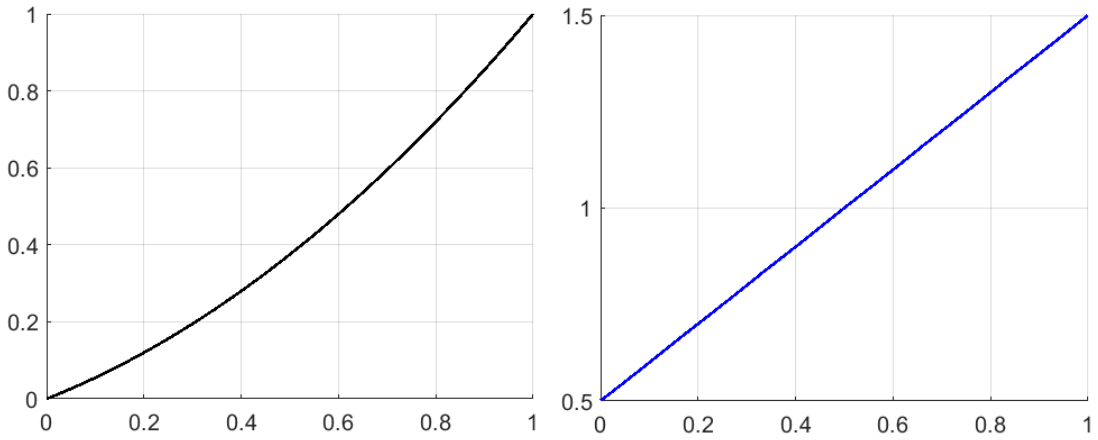
[11] presents an elementary example with a unique analytical solution. The problem is formulated as:

$$\begin{aligned} &min \;\; J = \int_0^1 \left(\frac{1}{2}u^2 + x\right) dt \\ &s.t.: \dot{x} = u, \qquad\qquad x(0) = 0, \;\; x(1) = 1 \end{aligned} \tag{5.18}$$

The optimality solution can be derived from the optimality necessary conditions (see [11] for detailed passages), as explained in Section 3.1.1, resulting in:

$$\begin{aligned} x^*(t) &= \frac{1}{2}(t^2 + t) \\ u^*(t) &= t + \frac{1}{2} \\ J^* &= \frac{23}{24} \approx 0.9583 \end{aligned} \tag{5.19}$$

The time interval is discretized with $m = 6$ mesh points, and the control is parameterized by a piecewise parabolic polynomial, not to impose the a priori knowledge on the optimal linear-control evolution. Using a trivial zero-control as initial guess, the variational multiple shooting is able to reproduce the analytical optimal solution, as depicted in Figure 5.9, with a final constraint violation of $10^{-10}$.




(a) 1D Test case state profile.

(b) 1D Test case control profile.

Figure 5.9: 1D test case optimal profiles.

Again, the derivative computation has been checked against the finite-difference approximations, and the results are reported in Table 5.4. The derivative check is completely passed and the verification phase on this example can be considered successfully completed.

| Derivative type | # Checked derivatives | Differing absolutely | Differing relatively | Differing both |
|---|---|---|---|---|
| DF | 24 | 0 | 0 | 0 |
| DG | 33 | 0 | 0 | 0 |
| HM | 60 | 0 | 0 | 0 |

Table 5.4: 1D test case derivative check.

The validation test, whose results are shown in Table 5.5, confirms again that the variational approach leads to a remarkable improvement in terms of computational time. The considerations on the contribution of the different derivative types made in Section 5.2.1 apply to this test case as well.

| Variational Derivative | J | Rel. CPU [-] | # NLP Iterations |
|---|---|---|---|
| DF-DG-HM | 0.983 | - | 6 |
| (a)DF-DG | 0.983 | 1.2 | 21 |
| (b)DF-DG | 0.983 | 8.5 | 6 |
| DF-HM | 0.983 | 2.0 | 6 |
| DG-HM | 0.983 | 3.0 | 6 |
| DF | 0.983 | 3.0 | 21 |
| DG | 0.983 | 4.0 | 21 |
| HM | 0.983 | 4.5 | 6 |
| none | 0.983 | 12.0 | 21 |

Table 5.5: 1D test case variational performance comparison.

### 5.2.3. FREE FINAL TIME TEST CASE

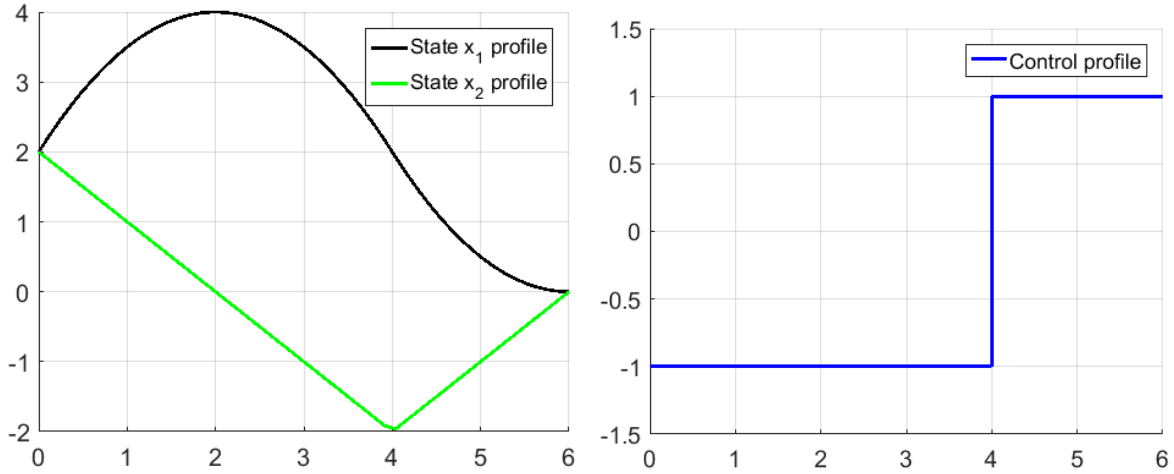
In the first two test cases the tool proved to compute the derivative information properly and to find the optimal solution accurately when a fixed-time controlled boundary-value-problem is investigated. This test case will verify the derivative computation and software implementation when time is a free parameter as well.

The test is formulated as a minimum-time problem to drive the state to the origin from a deviated condition:

$$\begin{aligned} &min \;\; J = \int_0^{t_f} dt \\ &s.t.: \dot{x}_1 = x_2, \qquad x_1(0) = 2, \;\; x_1(t_f) = 0 \\ &\qquad \dot{x}_2 = u, \qquad x_2(0) = 2, \;\; x_2(t_f) = 0 \\ &\qquad |u| \leq 1 \end{aligned} \tag{5.20}$$

where the path constraint $|u| \leq 1$ is another element to be tested. It is well known that the solution to this problem is a bang-bang profile with only one switch point as it is a linear second-order system [45]. The switch point is located at $t = 4$ s, and the final time, i.e. the objective index, results $t_f = 6$ s.

Using 6 sub-intervals and a piecewise linear polynomial control, with trivial zero-control as initial condition, the solution found with the variational multiple shooting is able to perfectly catch the bang-bang profile, as depicted in Figure 5.10, respecting the imposed path constraint.




(a) State profile.

(b) Control profile.

Figure 5.10: Free final time test case optimal profiles.

The derivative computation is checked and verified using WORHP's feature and the results are reported in Table 5.6.

| Derivative type | # Checked derivatives | Differing absolutely | Differing relatively | Differing both |
|---|---|---|---|---|
| DF | 25 | 0 | 0 | 0 |
| DG | 198 | 0 | 0 | 0 |
| HM | 61 | 0 | 0 | 0 |

Table 5.6: Free final time test case derivative check.

With the variational approach, the average computational time is approximately 0.15 s. By switching on and off the variational dynamics on different components, we have a deterioration in relative computational times as shown in Table 5.7.

| Variational Derivative | J | Rel. CPU [-] | # NLP Iterations |
|---|---|---|---|
| DF-DG-HM | 6.0 | - | 11 |
| (a)DF-DG | 6.0 | 1.5 | 30 |
| (b)DF-DG | 6.0 | 7.5 | 11 |
| DF-HM | 6.0 | 3.5 | 11 |
| DG-HM | 6.0 | 3.5 | 11 |
| DF | 6.0 | 4.5 | 30 |
| DG | 6.0 | 4.0 | 30 |
| HM | 6.0 | 5.0 | 11 |
| none | 6.0 | 9.5 | 30 |

Table 5.7: Free final time test case variational performance comparison.

Most of the considerations introduced in Section 5.2.1 still hold for this example as well. However, this test case introduces an interesting inversion in relative computational times between DF and DG computation by finite-difference approximations. Indeed, despite the group methods applied to computation of DG, the higher number of nonzero elements in the Jacobian (see Table 5.6) is predominant in worsening the performance, as we can see from the comparison of rows four with five and six with seven.

### 5.2.4. FREE FINAL STATE TEST CASE

The last test case run for verification is a free final state problem, a common scenario when multi-phase problems with discontinuous state linking conditions are studied, e.g. impulsive-maneuver trajectories. To analyze the behavior of the multiple-shooting implementation in this case, the elementary problem formulated as follows is studied.

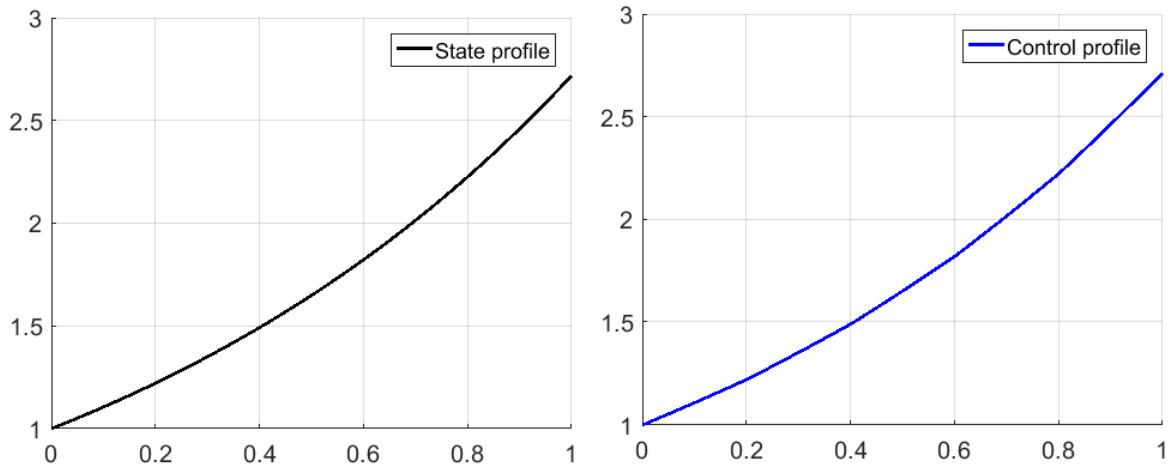
$$\begin{aligned} &min \;\; J = \int_0^1 (u-x)^2 dt \\ &s.t.: \dot{x} = u, \qquad\qquad x(0) = 1, \;\; x(1) = \text{free} \end{aligned} \tag{5.21}$$

As the final state is free, the optimal control profile is trivially $u^*(t) = x(t)$. Substituting in the dynamical equation and solving the simple differential equation with the given initial condition, the optimal state and control evolution is described by the exponential function:

$$u^*(t) = x^*(t) = e^t \tag{5.22}$$

Using 5 sub-intervals, a piecewise quadratic control parameterization, with the trivial zero-control initial guess, the variational multiple shooting manages to find the optimal solution, as in Equation (5.22), in 4 iterations and an average computational time of 0.01 s. The final state, at time $t_f = 1$, coincides with the Euler's number up to the 8th digit (final error of $10^{-8}$). The state and control evolution profiles are depicted in Figure 5.11.

The variational derivatives coincide with the finite-difference approximations under the set thresholds, as shown in Table 5.8.




(a) State profile. (b) Control profile.

Figure 5.11: Free final state test case optimal profiles.

| Derivative type | # Checked derivatives | Differing absolutely | Differing relatively | Differing both |
|---|---|---|---|---|
| DF | 20 | 0 | 0 | 0 |
| DG | 24 | 0 | 0 | 0 |
| HM | 50 | 0 | 0 | 0 |

Table 5.8: Free final state test case derivative check.

As can be seen from Table 5.9, the full variational approach is always the most efficient alternative. In this case, the number of nonzero elements in the gradient vector and the Jacobian matrix is really similar, while the Hessian matrix has more than double nonzero entries. For this reason, and because the number of iterations increases by a factor of four, the computation of HM by variational dynamics is the leading factor, rather than DF or DG, in this test-case.

| Variational Derivative | J | Rel. CPU [-] | # NLP Iterations |
|---|---|---|---|
| DF-DG-HM | 0.0 | - | 4 |
| (a)DF-DG | 0.0 | 2.0 | 17 |
| (b)DF-DG | 0.0 | 7.5 | 4 |
| DF-HM | 0.0 | 2.0 | 4 |
| DG-HM | 0.0 | 2.5 | 4 |
| DF | 0.0 | 3.5 | 17 |
| DG | 0.0 | 4.5 | 17 |
| HM | 0.0 | 3.0 | 4 |
| none | 0.0 | 7.5 | 17 |

Table 5.9: Free final state test case variational performance comparison.

## 5.3. SUMMARY

This chapter discussed the key features of the implemented variational multiple-shooting tool from a more practical point of view. The general flowchart has been explained and the different building blocks highlighted to have a better overview on what elements belong to the transcription method and what the interfaces are.

Successively, in the first section different practical details have been explained and the implementation choices motivated, such as the control parameterization through optimizable interpolating nodes, quadrature formulae, scaling routines, et cetera. The sparsity of the transcribed derivative matrices and its consequence in improving the efficiency of the corresponding NLP step have been explained. The next subsection showed how the employed NLP solver requires the sparsity information, and which communication logic of WORHP has been employed. Subsequently, scaling procedures have been analyzed and implemented to enhance the numerical stability of the code in the most generic case, still leaving enough flexibility to the user when the specific problem has to be set up. The last part of the section pointed out what types of problems the tool can handle, ranging from single to multi-phase problems (employing the linking conditions), from optimal continuous controls to impulsive ones.

The second section regarded the verification and validation process of the implemented tool. Initially, the strategies to verify and validate have been established and WORHP's feature employed for derivative comparison explained. Then, several elementary test-cases with different characteristics have been employed to test the variational multiple-shooting's functionality, confirming that the tool behaves properly on both boundary value or free boundary problems, on fixed and free-time cases, by managing to reproduce the known solution and checking the derivative computation. The validation routine confirmed that the variational approach actually results extremely efficient, leading to an improvement factor ranging from 7 to 12 with respect to the full finite-difference one.

# 6

# APPLICATIONS

The variational multiple-shooting tool has been implemented within the SMART repository [39] to be part of the optimization toolbox for solving practical aerospace problems. This chapter will present the algorithm application to two different test cases by employing other tools in SMART to generate the first-guess solution. This approach will serve a variety of goals. First, the interface to other software will be analyzed and improved. Second, the tool will be tested against large-scale problems, assessing its performance when a realistic example is run. Lastly, the selected test cases have a practical significance in the aerospace field and the variational multiple-shooting will be applied as a completed tool to investigate them.

The first section will discuss the feasibility study of a CubeSat rendezvous with an asteroid. This study involves the optimization of a low-thrust trajectory, a challenging and modern problem in space engineering. Along with that, it will enhance the limited literature about CubeSat rendezvous outside the Earth-Moon environment.

The second section will examine the application of the variational multiple-shooting tool to the 9th edition of the Global Trajectory Optimisation Competition (GTOC9) problem. First, the problem statement will be shortly discussed and the global strategy of the team outlined. Subsequently, the tool contribution to the proposed solution will be analyzed in detail.

## 6.1. CUBESAT RENDEZVOUS WITH ASTEROID

Several missions already proved that CubeSats can provide affordable access to space for research centres, small or medium companies and universities as well. For the latter, successful examples are Delfi-$C^3$ [46] and Delfi-n3Xt [47], the CubeSats missions part of the *Delfi Program* of Delft University of Technology [48].

Currently, the main research for CubeSats for deep space applications is carried out by big agencies. NASA recently selected the proposal Near-Earth Asteroid (NEA) Scout, a low-cost mission aiming to study a close asteroid and to demonstrate several technological innovations, such as being the first CubeSat to rendezvous an asteroid [49]. Similarly, ESA called proposals for an Asteroid Impact Mission (AIM) to the binary asteroid 65803 Didymos [50], having room for a total of six CubeSat units, having currently selected five of them before the final choice. A detailed analysis of advantages and technological difficulties of CubeSat's interplanetary applications can be found in [51].

The study presented in this section aims to demonstrate the feasibility of a low-thrust CubeSat-

asteroid rendezvous with the focus on orbital dynamics. The result of this analysis is of interest for future studies within the Aerospace Centre of Excellence (ACE), where the research has been carried out. The physical characteristics of the spacecraft and its low-thrust engine are summarized in the following table:

| Engine type | CubeSat units | Initial mass [kg] | Specific impulse [s] | Max Thrust [mN] |
|---|---|---|---|---|
| Low-thrust | 6 | 12 | 750 | 5 |

Table 6.1: CubeSat characteristics.

### 6.1.1. TARGET ASTEROID AND FIRST-GUESS GENERATION

The selection process of a suitable target asteroid has been carried out considering three essential factors:

- Time of flight (TOF) below 600 days;
- Departure date before year 2030;
- Limited propellant mass consumption.

As analyzing each NEA candidate singularly would have been time-consuming, the first selection has been performed using the *Center for Near Earth Object Studies (CNEOS)* database, created by the Jet Propulsion Laboratory (JPL) [52]. After restricting the choice to a handful of targets with Earth-like orbital elements and favourable angular position in the period 2020-2030, the selection continued running the spherical shaping tool (see Section 2.3.1). It was inserted in an outer loop defining a 3D uniform sampling grid (see Section 3.4.1), where the above defined time of flight is discretized using 8 days time steps, the departure date is discretized using 10 days time steps, and the number of revolutions allowed are between 0 and 2.

This led to the final pick of the asteroid $2000SG_{344}$, a small Aten asteroid with a diameter of 40 m and mass of $7.1 \times 10^7$ kg. It is among the top-10 near-Earth objects for impact likelihood with Earth [52], and NASA has been considering it for a manned sample return mission in 2069 [53]. The resulting pork-chop plot with the aforementioned time spans for $2000SG_{344}$ is depicted in Figure 6.1.

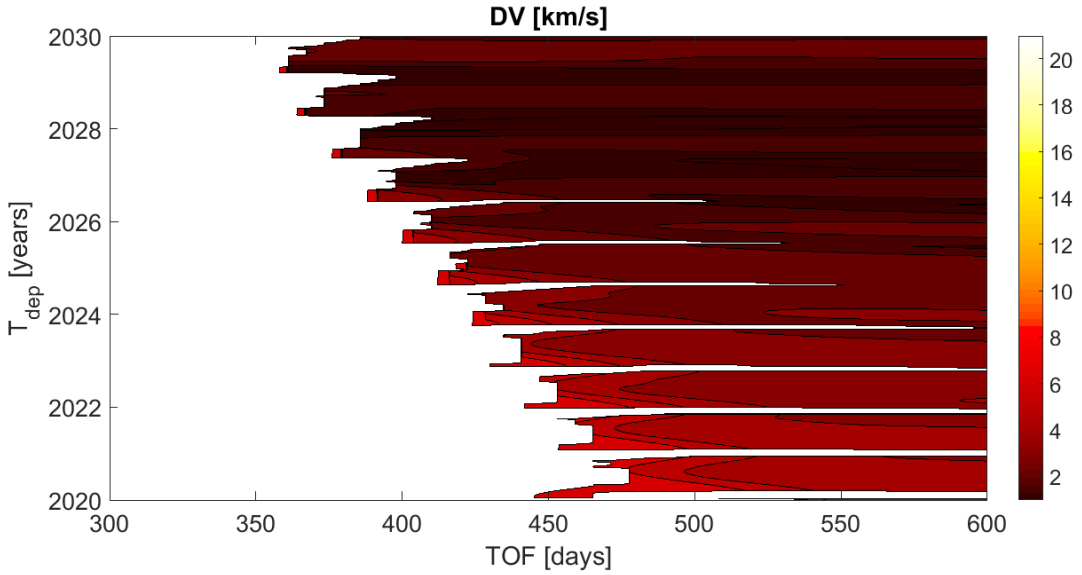



Figure 6.1: Pork-chop plot of Earth-$2000SG_{344}$ rendezvous.

Table 6.2 summarizes the most promising trajectories out of this analysis. In about 20 months the CubeSat manages to reach the asteroid following the cheapest trajectory (trajectory A), requiring only a reasonable portion of the initial mass as propellant. Further investigating the global search results, there are options more convenient in terms of time of flight and with comparable $\Delta V$ (e.g. trajectory B).

| Traj. ID | $T_{dep}$ [date] | TOF [days] | $\Delta V$ [km/s] | Final mass [kg] | TOF violation [days] |
|---|---|---|---|---|---|
| A | February 2027 | 595 | 0.976 | 10.509 | 0.5 |
| B | January 2028 | 392 | 0.978 | 10.505 | 0.4 |

Table 6.2: Earth-2000$SG_{344}$ most promising rendezvous from Spherical Shaping.

The TOF violation represents the discrepancy between the imposed time of flight and the actual one computed with the shaping. This would lead to a position and velocity mismatch between the asteroid and the final spacecraft position and velocity. This constraint violation shall be overturned by the multiple-shooting optimization.

### 6.1.2. CUBESAT RENDEZVOUS: MULTIPLE-SHOOTING OPTIMIZATION

The initial guess discussed in the previous section takes into account only the two-body pull caused by the Sun. However, the spacecraft departures from the Earth environment, where the Earth and Moon gravitational attraction is predominant. Thus, for the complete feasibility study, the gravity effects of three bodies (Sun, Earth and Moon) will be considered, using a *non-rotating heliocentric frame*. Therefore, the dynamical equations are:

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= -\frac{\mu_s}{r^3}\mathbf{r} - \mu_e \frac{\mathbf{r}-\mathbf{r_e}}{\|\mathbf{r}-\mathbf{r_e}\|^3} - \mu_m \frac{\mathbf{r}-\mathbf{r_m}}{\|\mathbf{r}-\mathbf{r_m}\|^3} + \mathbf{u} \\ \dot{m} &= -\frac{m \cdot u}{g_0 I_{sp}} \end{aligned} \tag{6.1}$$

where $\mu_s$, $\mu_e$ and $\mu_m$ are the standard gravitational parameters of the Sun, Earth and Moon respectively, whereas $\mathbf{r_e}$ and $\mathbf{r_m}$ are the position of the Earth and Moon with respect to the Sun, while $\mathbf{u}$ is the control acceleration and $u$ its magnitude.

The multiple shooting allows for any dynamical model, with the variational approach usable when the acceleration equations are at least twice differentiable. For Equation (6.1), the third-body variational dynamics can be derived in a similar fashion as was shown in Section 4.2.3. The variational multiple-shooting is initialized using the state evolution and linear approximation of the control extrapolated from the spherical shaping. However, since the Earth attraction is now considered, the initial state shall be slightly shifted to avoid singularity. Thus, the imposed initial state is the Earth-Moon Lagrangian point $L_2$. Along with the constraint on maximum thrust, two path constraints on minimum altitude over the Earth, i.e. $h_e \geq 400$ km, and over the Moon, i.e. $h_m \geq 50$ km, have been added to continue neglecting the Earth atmosphere reasonably and to avoid collisions with the ground. In order to keep using a linear parameterization of the control and still have a good optimality, the time-span has been subdivided into 100 segments, using a fixed step-size $RK8$ with 1-hour time-step as integration technique to compute a trajectory of high accuracy.

#### MINIMUM $\Delta V$ TRAJECTORY

The minimum $\Delta V$ trajectory (ID A in Table 6.2) optimized by the variational multiple shooting is depicted in Figure 6.2, showing the Earth and asteroid orbits as well, while Figure 6.3 illustrates the

resulting state evolution.

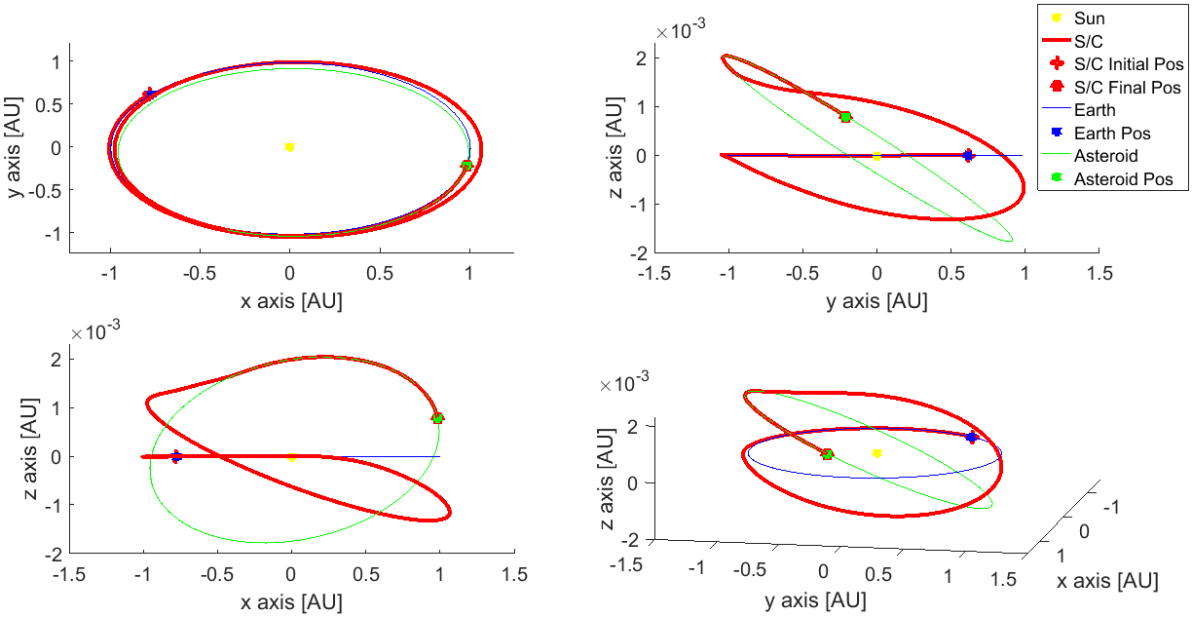



Figure 6.2: Optimal trajectory for CubeSat-Asteroid minimum $\Delta V$ rendezvous.

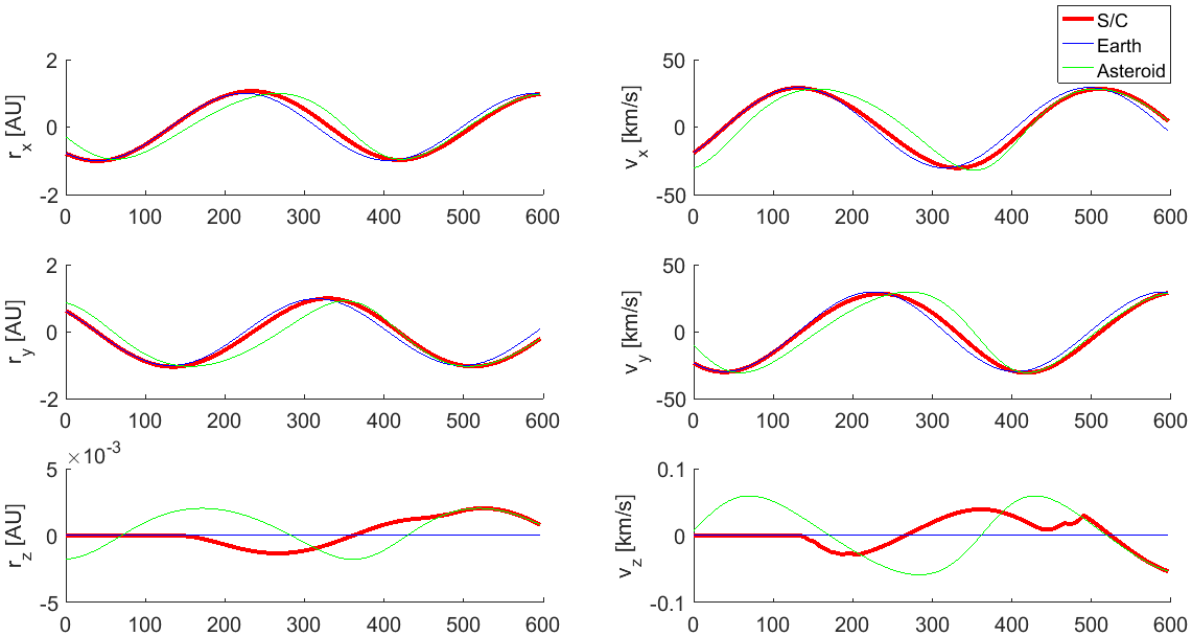



Figure 6.3: Optimal state profile for CubeSat-Asteroid minimum $\Delta V$ rendezvous.

As we can see from the trajectory evolution and the control profile shown in Figure 6.4, there are two main maneuvers where the orbital plane is changed and the orbit deformed properly. The last negligible adjustment, in the order of $10^{-5}$ N, corrects the velocity conditions to precisely match the asteroid state, as better shown in Figure 6.3.

The optimal solution benefits of two main thrust segments, equally distributed in time among all the control components, and three coasting periods. These arcs have been set to zero-thrust from the variational multiple shooting, as the initial guess from the spherical shaping did not include any

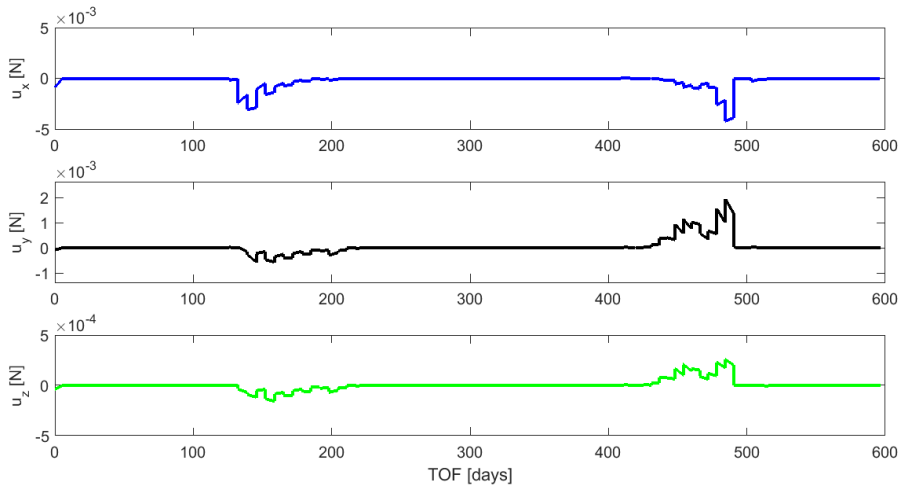



Figure 6.4: Optimal control profile for CubeSat-Asteroid minimum $\Delta V$ rendezvous.

coasting, as illustrated in Figure 6.5. Therefore, the refinement process transformed a continuous and low-peak control into a discontinuous profile with higher peaks.

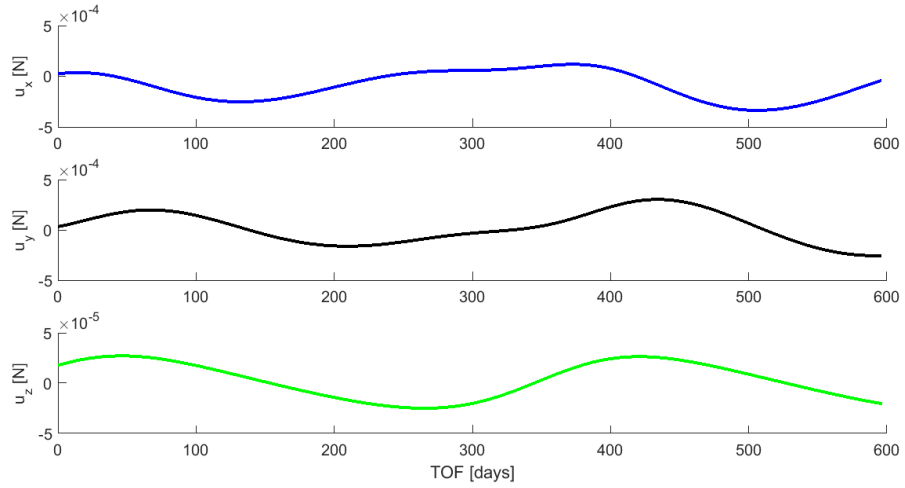



Figure 6.5: Initial control guess for CubeSat-Asteroid minimum $\Delta V$ rendezvous.

Reintegrating the equations of motion with the obtained optimal control, the final conditions are met up to $10^{-8}$ discrepancy resulting in a high-accuracy trajectory. The final mass has been lowered to 10.189 kg, i.e. 0.32 kg more of propellant mass with respect to the initial guess as discussed in Section 6.1.1. Although this does not seem a good result for an optimization algorithm, we shall keep in mind that the initial guess was generated neglecting the Earth and Moon attraction, and it was not respecting the imposed boundary conditions properly. Indeed, when the optimization with the variational multiple shooting is run with the Sun pull only, the final mass is 10.6 kg, resulting in a propellant saving of 0.1 kg. As a consequence, Earth and Moon are a burden for the CubeSat to escape the original orbit. This could be explained in two different ways: either the multiple shooting is not able to catch a gravity assist manoeuvre starting from an initial guess generated using the Sun attraction only, or the low level of thrust is not enough to enforce the necessary trajectory change for a fly-by.

### MINIMUM TOF TRAJECTORY

The second trajectory, the minimum TOF one (ID B in Table 6.2), is shown in Figure 6.6, where it is possible to notice that, in this case, the rendezvous is completed in just over one Earth revolution ($\approx$ 13 months).

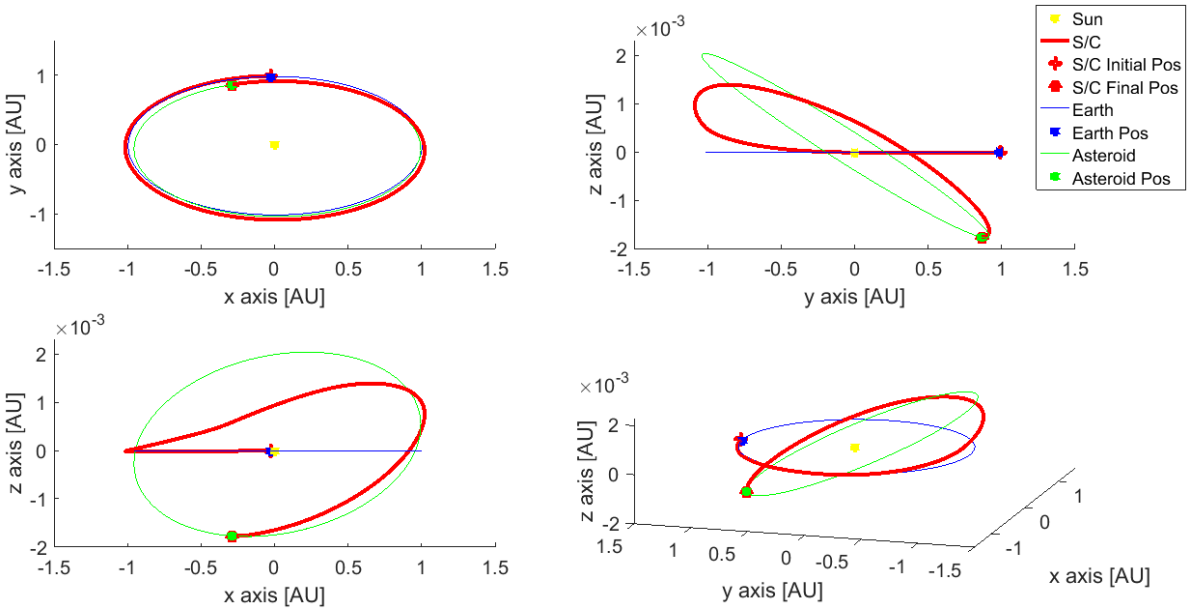



Figure 6.6: Optimal trajectory for CubeSat-Asteroid minimum time rendezvous.

The initial thrust segment, lasting three days, helps the spacecraft to slightly deviate from the Earth before the principal manoeuvre. From the control graph we can see a substantial difference with the minimum $\Delta V$ case. Only one main manoeuvre, before the final adjustment, can be detected, probably because the trajectory is shorter, while the final adjustment manoeuvre is more significant. This is confirmed by the plot of the control profile, as shown in Figure 6.7. Two main coasting arcs can be recognized, although some small thrust corrections are performed in the middle of the second one.

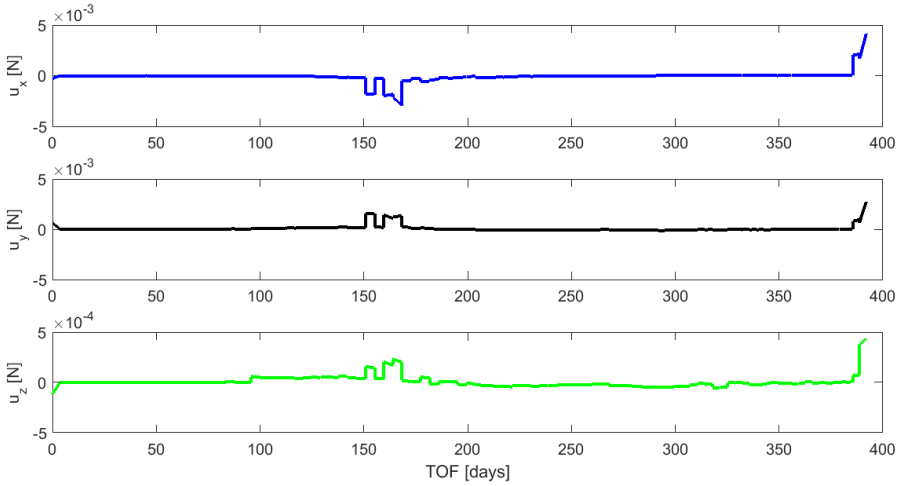



Figure 6.7: Optimal state profile for CubeSat-Asteroid minimum time rendezvous.

Figure 6.8 depicts the state evolution of the CubeSat, as well as the Earth and asteroid position and

velocity, and from this plot we can clearly see the effect of the final adjustment on the velocity component.

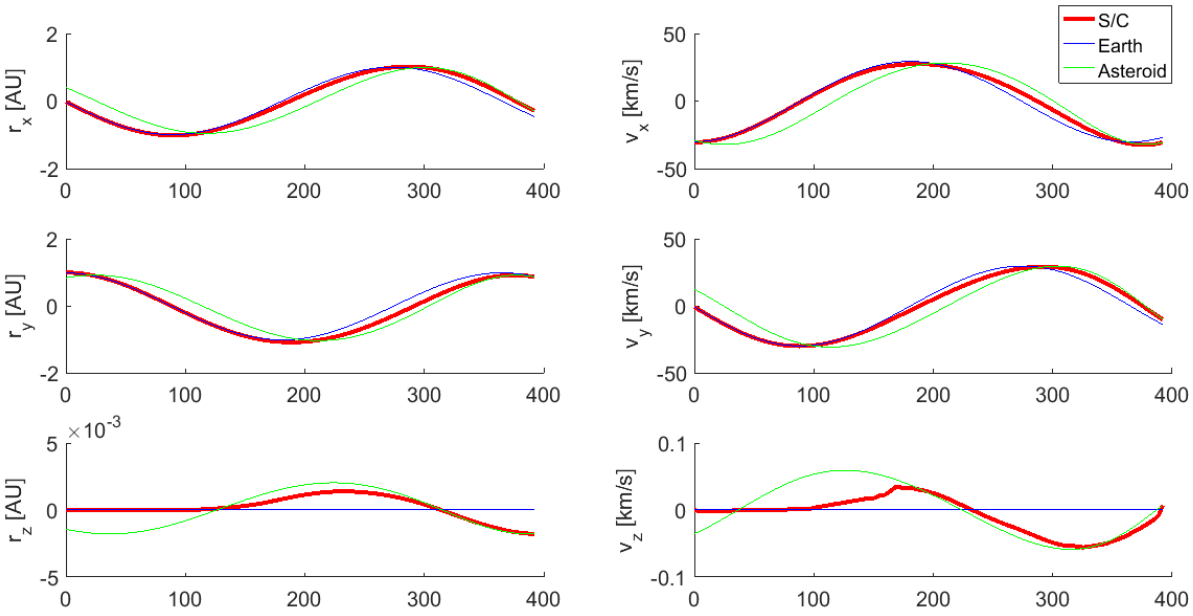



Figure 6.8: Optimal state profile for CubeSat-Asteroid minimum time rendezvous.

Also in this case, from comparison of the optimal control with the guessed one in Figure 6.9, we can see how the variational multiple shooting modifies the continuous profile from the shaping to a discontinuous one with higher peaks.

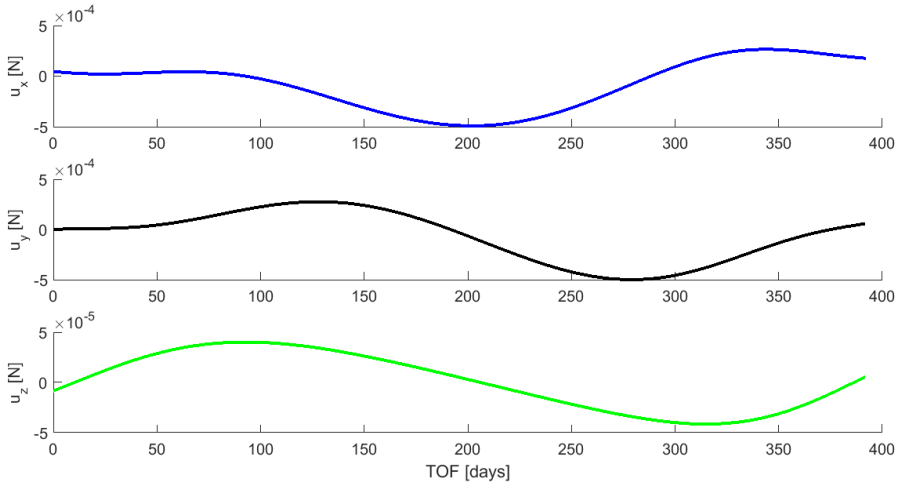



Figure 6.9: Initial control guess for CubeSat-Asteroid minimum time rendezvous.

When the trajectory is reintegrated for validation with the optimal control profile, the final conditions of the spacecraft and the asteroid coincides up to $10^{-8}$ again. The resulting final mass is 10.172 kg, i.e. 0.333 kg more of propellant mass with respect to the guessed one as reported in Section 6.1.1. The reasoning of this outcome explained for the minimum $\Delta V$ case holds for this trajectory as well.

### 6.1.3. CUBESAT RENDEZVOUS: CONCLUSIONS

For what concerns the discretized NLP sub-problems, associated to the cases discussed above, this low-thrust optimization example can be considered large-scale as it involved 1300 optimization parameters and 3706 constraints (more constraints than parameters as there are multiple nodes per interval to check the path constraints). Since the objective function does not depend on the position and velocity of the spacecraft, the sparsity pattern of the objective gradient has been used as well. The percentage of nonzero elements of the gradient vector is 53.85%, for the Jacobian matrix 1.01%, and for the Hessian matrix 1.06%. Thanks to the sparsity of the derivative matrix and the variational approach, the average computational time was lowered by an average factor of 6.0 with respect to the full finite-difference approach for both the cases. This factor resulted lower than the ones of the examples in Section 5.2 because, in this large-scale problem, a bigger share of the computational time is employed for the NLP step, which is the same regardless of the approach used to compute the derivatives.

The discussed analysis led to meaningful and interesting results, which are summarized in Table 6.3.

| Traj. ID | $T_{dep}$ [date] | TOF [days] | $\Delta V$ [km/s] | Final mass [kg] | Const. violation |
|---|---|---|---|---|---|
| A | February 2027 | 595 | 1.204 | 10.189 | $10^{-8}$ |
| B | January 2028 | 392 | 1.216 | 10.172 | $10^{-8}$ |

Table 6.3: Earth-2000$SG_{344}$ minimum $\Delta V$ and minimum TOF rendezvous after variational multiple-shooting.

The necessary propellant ratio is below a reasonable threshold, but this point shall be further assessed when successive studies on other subsystems will be available. The trajectory of minimum TOF seems the most convenient alternative, as it requires 200 days less than the other employing only 0.017 kg of propellant more.

This feasibility study is the first investigation of a rendezvous with the asteroid 2000$SG_{344}$ with low-thrust propulsion. In fact, all the previous studies analyzed only a high-thrust engine scenario, which involves bigger and more expensive spacecraft. As in the project presented in [53] NASA plans to send a crewed mission to the asteroid in 2069, it would be advantageous and effective to visit the asteroid with a low-cost CubeSat mission to perform reconnaissance and further studies well before the set date.

## 6.2. GTOC9

The Global Trajectory Optimisation Competition (GTOC) [54], also known as the America's cup of rocket science, is a worldwide month-long challenge initiated by the European Space Agency in 2005. The goal of the competition is to stimulate and advance research on optimization techniques for space mission applications.

### 6.2.1. Problem statement

Arrived at the 9th edition this year (GTOC9), the problem statement, named *The Kessler run* [55], requires to design a series of missions to de-orbit 123 debris objects which are gravely compromising the Sun-synchronous Low-Earth-Orbit (LEO) environment. A mission is modeled as a multiple-rendezvous trajectory which removes $N \geq 1$ debris thanks to de-orbit packages. The $i-th$ mission starts at a freely chosen debris and it ends when all the $N$ de-orbit packages have been released.

Following a rendezvous with a debris object, the de-orbit package takes 5 days of waiting time at the debris to complete its operations. After the debris remotion, the spacecraft can reach the next one only by impulsive maneuvers, i.e. instantaneous changes of the spacecraft velocity. The online validation system marks a rendezvous as valid if the final position and velocity of the spacecraft coincide with that of the debris under the given tolerances of 100 m and 1 m/s respectively. Constraints on the maximum propellant mass of a spacecraft and maximum waiting time between two rendezvous impose a multi-mission approach.

#### Spacecraft mass

The mass $m_{0_i}$ of the spacecraft at the beginning of the $i$th mission is $m_{0_i} = m_{dry} + Nm_{de} + m_p$. It is composed by a dry component, i.e. $m_{dry} = 2000$ kg, the de-orbit packages contribution, i.e. $Nm_{de}$ with $m_{de} = 30$ kg, and the propellant mass $m_p$ needed to control the spacecraft. The maximum propellant mass is 5000 kg.

#### Dynamics

Earth is modeled as an oblate planet up to the second spherical harmonics $J_2$. Therefore, the dynamics equations can be written in the *Earth-centered inertial frame* (ECI), defined in Section 2.1.1, as:

$$\begin{aligned}
\ddot{x} &= -\frac{\mu x}{r^3}\left[1+\frac{3}{2}J_2\left(\frac{r_{eq}}{r}\right)^2\left(1-5\frac{z^2}{r^2}\right)\right] \\
\ddot{y} &= -\frac{\mu y}{r^3}\left[1+\frac{3}{2}J_2\left(\frac{r_{eq}}{r}\right)^2\left(1-5\frac{z^2}{r^2}\right)\right] \\
\ddot{z} &= -\frac{\mu z}{r^3}\left[1+\frac{3}{2}J_2\left(\frac{r_{eq}}{r}\right)^2\left(3-5\frac{z^2}{r^2}\right)\right]
\end{aligned} \tag{6.2}$$

where $\mu$ is the Earth gravitational parameter, $r_{eq}$ the average equatorial radius of the Earth, and $r$ is the distance from Earth centre of mass, as already outlined in Section 2.2.2. To ensure this model to be reasonably realistic, a constraint on the minimum osculating periapsis has been imposed as $r_p \geq 6600$ km.

The mass evolves following the Tsiolkovsky rocket equation after each impulsive maneuver $\Delta V$:

$$m_f = m_i \cdot \exp\left(-\frac{\Delta V}{g_0 I_{sp}}\right) \tag{6.3}$$

where $g_0$ is the Earth gravitational force per unit of mass at sea level, and $I_{sp}$ = 340 s the engine specific impulse. A maximum of seven $\Delta V$s are allowed per transfer, two for departure and arrival, and five in-between two successive debris encounters.

#### OBJECTIVE FUNCTION

The cost function to be minimized is defined as follows:

$$J = \sum_{i=1}^{n} \left[ c_i + \alpha (m_{0_i} - m_{dry})^2 \right] \tag{6.4}$$

where $c_i$ is the base cost to launch the $i$-spacecraft in million of Euro (M€), $m_{0_i}$ its initial mass and $\alpha = 2 \cdot 10^{-6}$ [M€/kg$^2$] a scaling factor. The launcher cost increases linearly during the competition month, starting from 45 M€ to a maximum of 55 M€. Therefore, the objective function is composed by a linear term which favours complete campaigns with low number $n$ of missions, and a quadratic term preferring lighter spacecraft. The two terms represent conflicting goals, and the overall minimum shall be a good balance between them.

### 6.2.2. SOLUTION APPROACH

The variational multiple shooting has been applied to the GTOC9 challenge as part of the tools employed by the team Strathclyde++. The team approach was a three-step process (as depicted in the flowchart in Figure 6.10) :

- A combinatorial optimization step to generate multi-launch campaigns, performed using a Beam Search algorithm and a low-fidelity model for quick estimation of $\Delta V$. Among the output solutions generated with different heuristics, the most promising ones have been used as initial population for a multi-objective evolutionary optimization able to shuffle the debris sequence, the visiting times and the mass distribution between the launches;
- Global and local optimization of the impulses' magnitude and time of application within the single transfers using the differential evolution algorithm MP-AIDEA [56], which employs the Matlab solver *fmincon* to refine the minima after a set number of iterations. In this phase, out of the 1$e$6 function evaluations, 70% of them were using a non-expansive medium-fidelity dynamical model, while the other 30% the complete model to ensure a good initial guess for the next refinement step;
- Local optimization of multi-transfer launches, which was performed to respect the strict constraint tolerances and to further minimize the propellant mass of a mission. After a quick adjustment with a single shooting employing Matlab's *fmincon*, this local step is performed using the variational multiple shooting developed in the present thesis, to exploit its advantages in reducing the numerical integration errors and improve the convergence performance.

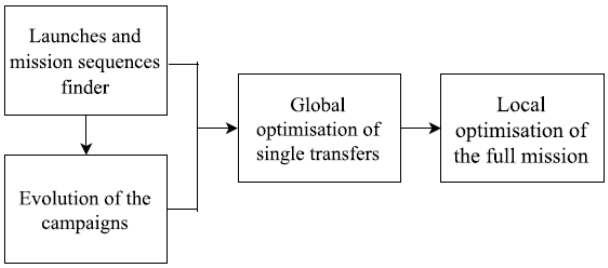



Figure 6.10: Flowchart of the solution approach [57].

The remainder of the section will focus on the variational multiple-shooting contribution to the proposed solution. For a complete and detailed explanation of the full approach of the team to the GTOC9, the paper [57] shall be consulted.

### 6.2.3. GTOC9: MULTIPLE-SHOOTING OPTIMIZATION

The optimization with the direct multiple-shooting has been set to handle a whole launch removing $N$ debris. The goal was to locally refine the solution coming from upper levels in the optimization cascade where either low-fidelity models, discrete time intervals or low/medium accuracy tools were used. As a consequence, usually the initial guess was not respecting the required final tolerance or it could be optimized further in terms of propellant mass.

Calling $\mathbf{y}$ the vector of free parameters, $t_i^d$ and $t_i^a$ the departure and arrival time to the $i$-th debris of the sequence, the constrained optimization problem for N consecutive transfers has been formulated as:

$$\begin{aligned} \min_{\mathbf{L}\le\mathbf{y}\le\mathbf{U}} \quad & J = \sum_{i=1}^{N-1}\sum_{j=1}^{n_{\Delta V}} \Delta V_{i_j} \\ \text{s.t.} \quad & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \\ & \mathbf{x}(t_i) = \mathbf{x}_{D_i}(t_i) \qquad\qquad i = 1,\dots,N \\ & a(1-e) \ge 6600km \\ & t_i^d \ge t_i^a + 5 \text{ days} \\ & t_{i+1}^a \le t_i^d + 25 \text{ days} \end{aligned} \tag{6.5}$$

where $\mathbf{x}_{D_i}(t_i)$ is the state of the $i$-th debris at $t_i$, and the last two constraints have been imposed by the problem formulation to simulate some control operation requirements. Given the nature of local refinement algorithms, this optimization step cannot handle the initial propellant mass constraint. Indeed, the latter is just a consequence of the $\Delta V$ necessary to complete the given rendezvous sequence, a quantity that cannot be lowered ad libitum.

Each transfer between two debris objects is modeled as a multi-phase problem with discontinuous linking conditions (see Section 5.1.5), i.e. the instantaneous velocity change $\Delta V$. In the single phase, there is no continuous control to optimize and also a single discretization interval could be used. Nonetheless, $m$ sub-intervals have been introduced to reduce the integration errors and to enhance the numerical solution of the boundary value problem, restoring the original purpose of shooting techniques [21]. In particular, this precaution was necessary because a single transfer could last up to 25 days, which translates in hundreds of revolutions in the fast LEO dynamics under the effect of the full $J_2$ disturbance. Hence, the number of free parameters per transfer sums up to $n_{y_i} = 4n_{\Delta V} + 6(n_{\Delta V}-1)(m-1)$, where the first term describes the time and three vector components of the impulsive maneuvers, while the second term concerns the initial condition of each sub-interval. Successively, each transfer is connected to the next one by means of a coasting stage, i.e. the de-orbit phase at the debris, with continuous linking conditions.

To employ the variational approach, the dynamics discussed in Section 6.2.1 shall be enhanced with the associated first- and second-order variational dynamics. Exploiting the distributive property of derivatives, the variational dynamics associated to the term describing the central-body pull in Equation (6.2) can be derived as shown in Section 4.2.3. On the contrary, the equations describing the sensitivity evolution caused by the $J_2$ effect shall be derived ex novo. Following the procedures explained in Section 4.2, the variational equations are derived using the symbolic mathematical and analytical software Maple 2016 (see Appendix A for $J_2$ Jacobian and Hessian matrices).

As settings, $m = 11$ grid discretization points per phase and an RK4 integrator with 120 s time-step were used. This translates in about 400 free variables per transfer, while the number of complete free parameters depends on how many rendezvous are performed in one mission. Throughout the competition month, different solutions were optimized with the variational multiple shooting, however only the final submitted one will be analyzed in this section as the considerations that will be presented hold as a general rule for this kind of problem and settings.

Table 6.4 summarizes the improvement in initial mass for each mission in the final submitted campaign when starting from the initial guess given by the previous single shooting step. The saved mass is mainly related to the number of debris objects removed in a specific mission.

| Launch | N. debris | Init. mass [kg] | Opt. mass [kg] | Saved mass [kg] |
|---|---|---|---|---|
| 1 | 2 | 2527.54 | 2519.82 | 7.72 |
| 2 | 8 | 3640,16 | 3631,14 | 9.02 |
| 3 | 9 | 4988.12 | 4975.13 | 12.99 |
| 4 | 5 | 3163.28 | 3158.84 | 4.44 |
| 5 | 4 | 3330.31 | 3326.34 | 3.97 |
| 6 | 5 | 4500.42 | 4491.43 | 8.99 |
| 7 | 7 | 4753.69 | 4747.27 | 6.41 |
| 8 | 17 | 5322.51 | 5304.57 | 17.97 |
| 9 | 15 | 5208.94 | 5188.77 | 20.17 |
| 10 | 16 | 5601.07 | 5583.23 | 17.84 |
| 11 | 10 | 4304.23 | 4290.82 | 13.41 |
| 12 | 9 | 3486.32 | 3476.51 | 9.81 |
| 13 | 9 | 3649.43 | 3637.31 | 12.12 |
| 14 | 7 | 3481.65 | 3478.57 | 3.07 |
| Total | 123 | 57957.66 | 57809.75 | 147.25 |

Table 6.4: GTOC9 Strathclyde++ final solution improvement with variational multiple-shooting.

The variational multiple shooting enhanced significantly the campaign efficiency in terms of propellant consumption. This improvement reflected in a decrease of about 2 M€ for the total debris-removal campaign cost. In a number of cases, the initial guess generated by the previous optimization steps did not manage to pass the online validation check because of the final state threshold and/or the minimum pericenter constraint. The multiple shooting, through the enforcement of proper path and boundary constraints with low thresholds, managed to revert these constraint violations and produce high-accurate trajectory always able to pass the validation check.

For what concerns the discretized NLP associated problem, the scale ranges from medium to large depending on the number of transfers for a given launch. For example, mission 1, which removes only two debris objects, employs about 400 free parameters, while the discretization of mission 8, de-orbiting 17 debris, has a total of about 6400 optimizable variables. The number of sub-intervals per phase, along with the high number of stages in the complete multi-phase problem, led to very sparse derivative matrices. Indeed, both the Jacobian and Hessian nonzero elements' percentages are under 0.1%, as it can be seen in Figure 6.11 where an illustrative and scaled pattern of a single transfer has been depicted. The secondary diagonal on the right-hand side of the plot represents the nonzero derivatives with respect to the time of the impulses. As the objective function depends only on the $\Delta V$, the percentage of nonzero elements can be computed analytically as $DF_{nnz} = 3n_{\Delta V}/n_{y_i} = 3n_{\Delta V}/[4n_{\Delta V} + 6(n_{\Delta V} - 1)(m-1)]$, approximately 5% given the settings as above.

The variational approach leads to an average gain of 6.5-7.0 in CPU-time, depending on the number of rendezvous in the mission, which determines the number of phases. In this example,

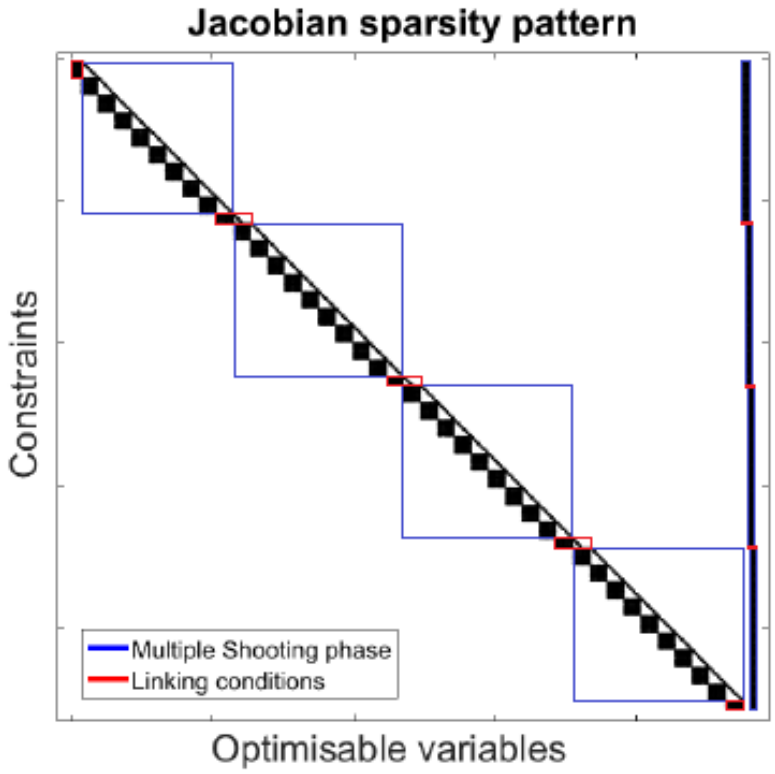



Figure 6.11: Scaled Jacobian sparsity pattern of single GTOC9 rendezvous transfer with 5 $\Delta V$.

the variational approach results to be more effective than the case discussed in Section 6.1 because the associated NLP subproblem is averagely smaller, and therefore it takes a lower percentage of the total computational time. Indeed, it involves sparser Gradient, Jacobian and Hessian quantities for comparable orders of free variables.

### 6.2.4. GTOC9: Conclusions

Thanks to the GTOC9, the variational multiple shooting has been validated against a novel problem scenario which challenged 69 teams worldwide. As shown in the previous section, the algorithm developed in this thesis proved to be crucial to make the solution satisfying the required constraints within the set threshold, and as a consequence to make the final campaign valid for the online check tool. By means of discussion on the GTOC forum [58] and at the GTOC special session at the 31st ISTS, 26th ISSFD & 8th NSAT Joint Conference in Japan, where the paper [57] was presented, it emerged that teams using only a single shooting method had convergence issues and solutions rejected from the validation tool. These debates further confirm the advantages of the employed multiple shooting algorithm and its compatibility with the problem dynamics. Furthermore, the tool resulted beneficial in terms of objective function optimization as well, as it manages to lower the propellant mass considerably. The variational dynamics managed to speed-up the computation burden dramatically, giving an additional validation of the usefulness of this technique.

Thanks to the solution approach discussed in Section 6.2.2, the team Strathclyde++ ranked $6^{th}$ in the competition, as it is reported in Figure 6.12 showing the score of the teams who submitted valid solutions.

| Rank | Team Name | Missions | Removed | $J$ in MEUR |
|---|---|---|---|---|
| 1 | Jet Propulsion Laboratory | 10 | 123 | 731.2756 |
| 2 | NUDT Team | 12 | 123 | 786.21452 |
| 3 | XSCC-ADL | 12 | 123 | 821.37966 |
| 4 | Tsinghua-LAD | 12 | 123 | 829.57987 |
| 5 | NPU | 13 | 123 | 878.99821 |
| 6 | Strathclyde++ | 14 | 123 | 918.9808 |
| 7 | DLR | 14 | 123 | 949.85389 |
| 8 | Missions Learners | 14 | 123 | 964.51134 |
| 9 | The Aerospace Corporation | 14 | 123 | 1004.4860 |
| 10 | Team Jena | 15 | 123 | 1022.9063 |
| 11 | UT Austin | 15 | 122 | 1044.1787 |
| 12 | NJU Team | 16 | 123 | 1047.9685 |
| 13 | EFLMAN TEAM | 14 | 119 | 1107.6936 |
| 14 | CU Boulder | 17 | 123 | 1150.8439 |
| 15 | CAS-NUAA | 14 | 123 | 1182.0632 |
| 16 | MTU-UoM | 16 | 122 | 1192.7433 |
| 17 | NSSC-THU | 16 | 122 | 1210.3333 |
| 18 | Brute WORHP | 18 | 123 | 1229.5475 |
| 19 | The Goonies | 15 | 122 | 1238.6396 |
| 20 | NablaZeroLabs | 16 | 123 | 1267.7501 |
| 21 | TYSE | 16 | 123 | 1336.8590 |
| 22 | TM | 18 | 123 | 1490.9659 |
| 23 | Occitania | 22 | 120 | 1493.8567 |
| 24 | ARGoPS | 20 | 123 | 1512.6017 |
| 25 | Personal team | 23 | 123 | 1588.5770 |
| 26 | GO to space | 20 | 112 | 1819.1391 |
| 27 | UofI and Goddard | 23 | 123 | 1951.6797 |
| 28 | LSPirates | 20 | 105 | 2164.2321 |
| 29 | Astro-ASAP-UC3M | 13 | 85 | 3141.1951 |
| 30 | Cal Poly SLO | 39 | 84 | 4467.8746 |
| 31 | Team STAR Lab | 12 | 57 | 4481.7781 |
| 32 | Nicolas RAVE | 13 | 18 | 6453.0254 |
| 33 | National University of Colombia | 2 | 7 | 6511.5471 |
| 34 | MeltedCode | 1 | 5 | 6594.1105 |
| 35 | AMSS_GTOC | 1 | 4 | 6619.3569 |
| 36 | Bremen optimizers | 1 | 2 | 6760.20 |

Figure 6.12: Top 15 solution rankings for the GTOC9 [54].

# 7

# CONCLUSIONS

This chapter, which concludes the thesis report, has the purpose to summarize and present the research results from a broader picture, with a particular focus on the contributions yielded to the related optimization literature. In addition, ideas for possible future improvements or research developments of the topic will be suggested.

## 7.1. RESEARCH OUTCOMES

The objective of this thesis research was the development of a numerical optimization tool using the propagation of the associated variational dynamics to compute the derivative information. This variational approach was the major novelty element, and therefore has been the center of theoretical development, analysis and assessment. In order to perform this investigation, theoretical background information about orbital mechanics and optimization theory were needed. The former, analyzed in Chapter 2, constituted a useful summary of astrodynamics topics and tools which have been used at later stages in the thesis. For example, the reported perturbation mathematical models were beneficial for the examination of the space related examples' dynamics and results, while the discussed inverse method has been used for the first-guess generation of a low-thrust interplanetary trajectory. On the other hand, the optimization theory, examined in Chapter 3, represented a crucial study for the transcription method selection and for identifying the potential usefulness of the variational approach. Once these background elements had been reviewed and critically examined, three major novel contributions have been introduced in the central part of the thesis:

- In Chapter 4, the theory of variational equations for optimization applications has been formulated in detail, expanding considerably the preceding concepts available in literature. In particular, the second-order variational equations are a novel mathematical development, a result of the author's personal contribution;
- The analysis of the variational approach performance, another almost overlooked topic in literature, has been carried out between Chapters 5 and 6. This novel technique proved to enhance dramatically the efficiency of multiple-shooting algorithms by speeding-up the derivative computation up to one order of magnitude less for some applications. Both elementary test cases with known solutions and complex large-scale optimization problems have been used for validation and performance assessment, and the results' analyses have always supported the improved efficiency and the high-accuracy of the variational multiple shooting tool;

- The optimization applied to the complex test cases in Chapter 6 led to interesting and challenging results. For what concerns the CubeSat-asteroid rendezvous, this phase 0 study led to a feasible trajectory with reasonable propellant mass consumption. Moreover, the 2028 trajectory to the NEA $2000SG_{344}$ could be of interest for an actual low-cost asteroid reconnaissance mission preceding the proposed 2069 manned mission investigated by NASA. Regarding the *Kessler Run* application, the variational multiple shooting yields to the saving of a huge portion of propellant mass, in particular if we consider that the initial guess was already generated using a single-shooting method. Furthermore, it managed to generate high-accuracy trajectories able to respect the imposed strict constraint thresholds, where other methods failed. The developed tool helped the team Strathclyde++ to finish $6^{th}$ in a worldwide renowned challenge as the GTOC.

## 7.2. FUTURE DEVELOPMENTS

The variational multiple shooting tool developed in this thesis still has room for improvements. Interesting future developments could be the substitution of the multiple shooting with a *parallel shooting* [3], an algorithm which propagates the quantities corresponding to independent shooting sub-intervals on different processors of the desktop hardware. Although WORHP already employs parallel routines for the nonlinear programming step, these parallel processes would not be in conflict as the propagation phase and the NLP sub-routines are consecutive non-simultaneous steps.

Another possible future study would be the characterization of the variational approach performance with different NLP algorithms, to assess which solver is more suitable to be interfaced with this technique. Indeed, different NLPs employ different heuristics, group methods for finite-difference derivative computation, and different algorithms for the solution of the nonlinear constrained optimization problem.

Additionally, it would be beneficial to generalize the tool to work on non-sequential multi-phase problems, while currently it is designed only for phases sequential in time. This generalization would allow the optimization tool to take into account simultaneous dynamics and/or constraints. For example, in launcher trajectory optimization the constraint on a stage impact point is concurrent with the other stages' course optimization. For space applications, an interesting case could be the simultaneous optimization of probe and lander trajectories after their separation.

Lastly, the development of a hybrid tool, as discussed in Section 3.2.4, would be an interesting field of research, in order to exploit the advantages of both collocation and shooting techniques, reduce the dependency on external tools to generate initial guesses and improve the computational performance and numerical stability.

# BIBLIOGRAPHY


[1] K. Alemany and R. D. Braun, *Survey of global optimization methods for low-thrust, multiple asteroid tour missions,* in *Advances in the Astronautical Sciences*, Vol. 127 PART 2 (2007) pp. 1641–1660.

[2] S. Kemble, *Interplanetary mission analysis and design*, 1st ed. (Springer, 2006).

[3] J. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming*, 1st ed. (Society for Industrial & Applied Mathematics, 2001).

[4] K. F. Wakker, *Fundamentals of Astrodynamics* (Institutional Repository TU Delft, 2015).

[5] E. Mooij, *Re-entry Systems*, Tech. Rep. (Delft University of Technology, 2015).

[6] R. Baker, *Astrodynamics, applications and advanced topics* (Academic Press Inc. New York, 1967).

[7] W. Kaula, *Theory of Satellite Geodesy* (Blaisdell Publishing Company, 1966).

[8] D. Izzo, *Lambert's problem for exponential sinusoids,* Journal of Guidance, Control, and Dynamics **29**, 1242 (2006), https://doi.org/10.2514/1.21796.

[9] A. E. Petropoulos, J. M. Longuski, and N. X. Vinh, *Shape-based analytic representations of low-thrust trajectories for gravity-assist applications,* Advances in the Astronautical Sciences **103**, 563 (2000).

[10] D. M. Novak and M. Vasile, *Improved shaping approach to the preliminary design of low-thrust trajectories,* Journal of Guidance, Control, and Dynamics **34**, 128 (2011), https://doi.org/10.2514/1.50434.

[11] H. G. Visser, *Aircraft Performance Optimization*, Tech. Rep. (Delft University of Technology, 2014).

[12] H. Goldstein, *Classical Mechanics*, 3rd ed. (Pearson, 2001).

[13] G. Bliss, *Lectures on the Calculus of Variations*, 1st ed. (University of Chicago Press, 1946).

[14] B. A. Conway, *Spacecraft Trajectory Optimization* (Cambridge University Press, New York, 2010).

[15] R. H. Bishop and D. M. Azimov, *Analytical space trajectories for extremal motion with low-thrust exhaust-modulated propulsion,* Journal of Spacecraft and Rockets **38**, 897 (2001).

[16] J. A. Kechichian, *Optimal low-thrust transfer using variable bounded thrust,* Acta Astronautica **36**, 357 (1995), https://doi.org/10.1016/0094-5765(95)00112-3.

[17] A. Bryson and Y. Ho, *Applied Optimal Control* (John Wiley & Sons, 1975).

[18] J. Betts, *Survey of numerical methods for trajectory optimization,* Journal of Guidance, Control and Dynamics **21** (1998), https://doi.org/10.2514/2.4231.

[19] O. Brüls, G. Bastos Jr., and R. Seifried, *A stable inversion method for feedforward control of constrained flexible multibody systems,* Journal of Computational and Nonlinear Dynamics **9** (2014), https://doi.org/10.1115/1.4025476.

[20] F. Zuiani and M. Vasile, *Direct transcription of low-thrust trajectories with finite trajectory elements,* in *61st International Astronautical Congress, Prague, CZ* (2010).

[21] P. F. Gath, *CAMTOS - A Software Suite Combining Direct and Indirect Trajectory Optimization Methods*, Master thesis, University of Stuttgart (2002).

[22] P. Gill, W. Murray, and M. Wright, *Numerical Linear Algebra and Optimization* (Perseus Books, 1990).

[23] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. (Wiley and Sons, New York, 1985).

[24] P. Gill, W. Murray, and M. Wright, *Practical optimization*, 1st ed. (Academic Press, London, 1981).

[25] A. Forsgren, P. Gill, and M. Wright, *Interior methods for nonlinear optimization,* SIAM review **44-4**, 525–597 (2002).

[26] *Users' Guide to WORHP 1.10*, Steinbeis-Forschungszentrum Optimierung, Steuerung und Regelung (2017), https://worhp.de/latest/download/user_manual.pdf.

[27] T. Linke, D. Wassel, and C. Büskens, *Recent advances in the solution of large nonlinear optimisation,* in *Engineering Optimization IV*, edited by H. Rodrigues, J. Herskovits, C. M. Soares, and J. M. Guedes (Taylor & Francis, 2014) pp. 141–146.

[28] J. T. Olympio, *Optimal control problem for low-thrust multiple asteroid tour missions,* Journal of Guidance, Control, and Dynamics **34**, 1709 (2011), https://doi.org/10.2514/1.53339.

[29] P. De Pascale and M. Vasile, *Preliminary design of low-thrust multiple gravity-assist trajectories,* Journal of Spacecraft and Rockets **43**, 1065 (2006), https://doi.org/10.2514/1.19646.

[30] T. McConaghy, T. Debban, A. Petropoulos, and J. Longuski, *Design and optimization of low-thrust trajectories with gravity assists,* Journal of Spacecraft and Rockets **40** (2003).

[31] S. Rao, *Engineering Optimization*, 4th ed. (Blaisdell publishing company, 2009).

[32] D. Maringer, *Portfolio Management with Heuristic Optimization* (Springer, 2005).

[33] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical Analysis*, 10th ed. (Cengage Learning, 2015).

[34] R. Fletcher, *Practical methods of optimization*, 2nd ed. (Wiley, 2000).

[35] L. Biegler, J. Nocedal, and C. Schmid, *A reduced Hessian method for large-scale constrained optimization,* SIAM Journal on Optimization **5**, 314–347 (1995).

[36] C. Kirches, *A Numerical Method for Nonlinear Robust Optimal Control with Implicit Discontinuities and an Application to Powertrain Oscillations*, Ph.D. thesis, Heidelberg University (2006).

[37] R. Quirynen, *Automatic code generation of Implicit Runge-Kutta integrators with continuous output for fast embedded optimization*, Master thesis, Katholieke Universiteit Leuven (2012).

[38] C. Jänsch, K. Schnepper, and K. H. Well, *Multi-phase trajectory optimization methods with applications to hypersonic vehicles,* in *Applied Mathematics in Aerospace Science and Engineering*, edited by A. Miele and A. Salvetti (Springer, 1994) Chap. 8, pp. 133–164.

[39] Mechanical and Aerospace Engineering Department of Strathclyde University, *SMART - Strathclyde Mechanical and Aerospace Research Toolbox,* https://github.com/strath-ace (2016).

[40] H. B. Bock, *Numerical treatment of inverse problems in chemical reaction kinetics,* in *Modelling of Chemical Reaction Systems*, Springer Series in Chemical Physics, Vol. 18, edited by K. H. Ebert, P. Deuflhard, and W. Jäger (Springer-Verlag, 1981) pp. 102–125.

[41] K. Brenan, *Engineering Methods Report: The Design and Development of a Consistent Integrator/Interpolator For Optimization Problems*, Aerospace Technical Memorandum ATM 88-52 (The Aerospace Corporation, 1988).

[42] C. W. Gear and T. Vu, *Smooth numerical solutions of ordinary differential equations,* in *Workshop on Numerical Treatment of Inverse Problems for Differential and Integral Equations* (Heidelberg, 1982).

[43] T. Vu, *Numerical Methods for Smooth Solutions of Ordinary Differential Equations*, Tech. Rep. (Department of Computer Science - University of Illinois, 1983).

[44] H. Bock and K. Plitt, *A multiple shooting algorithm for direct solution of optimal control problems,* in *9th World Congress*, Vol. IX (International Federation of Automatic Control, 1984).

[45] X. Wang, *Solving optimal control problems with MATLAB - Indirect methods*, Tech. Rep. (ISE Dept. of North Carolina State University, 2009).

[46] A. Bonnema, W. J. Ubbels, J. Rotteveel, E. van der Linden, and E. van Breukelen, *Delfi-C3: Technology demonstrator mission at Delft University of Technology,* in *STEC (Space Technology Education Conference)* (Aalborg University, Aalborg, Denmark, 2005).

[47] S. de Jong, E. Maddox, G. J. Vollmuller, C. A. H. Schuurbiers, R. A. C. M. M. van Swaaij, W. J. Ubbels, and R. J. Hamann, *The Delfi-n3Xt Nanosatellite: Space Weather Research and Qualification of Microtechnology,* in *Proceedings of the 59th IAC (International Astronautical Congress)* (Glasgow, Scotland, UK, 2008).

[48] R. Hamann, C. Verhoeven, A. Vaartjes, and A. Bonnema, *Nanosatellites for microtechnology pre-qualification: The Delfi program of Delft University of Technology,* in *6th Symposium on Small Satellites for Earth Observation* (Berlin, Germany, 2007).

[49] L. Johnson, J. Castillo-Rogez, J. Dervan, and L. McNutt, *Near-Earth Asteroid (NEA) Scout,* in *AIAA SPACE 2014 Conference and Exposition* (San Diego, CA, 2016).

[50] F. Ferrari, M. Lavagna, I. Gerth, B. Burmann, M. Scheper, and I. Carnelli, *ESA's Asteroid Impact Mission: mission analysis and payload operations state of the art,* in *ESA INDICO* (San Diego, CA, 2016).

[51] R. Staehle, B. Anderson, B. Betts, D. Blaney, C. Chow, L. Friedman, H. Hemmati, D. Jones, A. Klesh, P. Liewer, J. Lazio, M. W.-Y. Lo, P. Mouroulis, N. Murphy, P. J. Pingree, J. Puig-Suari, T. Svitek, A. Williams, and T. Wilson, *Interplanetary CubeSats: Opening the Solar System to a Broad Community at Lower Cost*, Tech. Rep. (NASA Office of the Chief Technologist, 2012).

[52] Jet Propulsion Laboratory, *Center for Near Earth Object Studies website,* https://cneos.jpl.nasa.gov/ (2017).

[53] D. J. Korsmeyer, R. R. Landis, and P. A. Abell, *Into the beyond: A crewed mission to a near-earth object,* in *International Astronautical Congress* (Valenica, Spain, 2006).

[54] D. Izzo, *GTOC portal,* https://sophia.estec.esa.int/gtoc_portal/ (2017).

[55] D. Izzo, *Problem description for the 9th Global Trajectory Optimisation Competition,* https://doi.org/10.5281/zenodo.570193 (2017).

[56] M. Di Carlo, M. Vasile, and E. Minisci, *Multi-population inflationary differential evolution algorithm with adaptive local restart,* in *2015 IEEE Congress on Evolutionary Computation, CEC 2015 - Proceedings* (2015) pp. 632–639, https://doi.org/10.1109/CEC.2015.7256950.

[57] C. O. Absil, L. A. Ricciardi, M. Di Carlo, C. Greco, R. Serra, M. Polnik, A. Vroom, A. Riccardi, E. Minisci, and M. Vasile, *On the generation and evolution of multiple debris removal missions,* in *31st ISTS, 26th ISSFD & 8th NSAT Joint Conference* (Matsuyama, Japan, 2017) https://sophia.estec.esa.int/gtoc_portal/wp-content/uploads/2017/05/gtoc9-strath.pdf.

[58] D. Izzo, *GTOC Discussion Forum,* https://kelvins.esa.int/gtoc9-kessler-run/discussion/ (2017).

# A

# $J_2$ VARIATIONAL DYNAMICS

Jacobian and Hessian matrices associated to variational dynamics of the $J_2$ oblateness perturbation.

$$
\begin{bmatrix}
0 & 0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0\,0\,0 \\
\frac{3\mu J2 req^2\left(4x^4+3x^2y^2-27x^2z^2-y^4+3y^2z^2+4z^4\right)}{2\left(x^2+y^2+z^2\right)^{9/2}} & \frac{15\mu y J2 req^2 x\left(x^2+y^2-6z^2\right)}{2\left(x^2+y^2+z^2\right)^{9/2}} & \frac{45z\left(x^2+y^2-\frac{4z^2}{3}\right)x\mu J2 req^2}{2\left(x^2+y^2+z^2\right)^{9/2}} & 0\,0\,0 \\
\frac{15\mu y J2 req^2 x\left(x^2+y^2-6z^2\right)}{2\left(x^2+y^2+z^2\right)^{9/2}} & -\frac{3\mu J2 req^2\left(x^4-3x^2y^2-3x^2z^2-4y^4+27y^2z^2-4z^4\right)}{2\left(x^2+y^2+z^2\right)^{9/2}} & \frac{45z\left(x^2+y^2-\frac{4z^2}{3}\right)\mu J2 y req^2}{2\left(x^2+y^2+z^2\right)^{9/2}} & 0\,0\,0 \\
\frac{45z\left(x^2+y^2-\frac{4z^2}{3}\right)x\mu J2 req^2}{2\left(x^2+y^2+z^2\right)^{9/2}} & \frac{45z\left(x^2+y^2-\frac{4z^2}{3}\right)\mu J2 y req^2}{2\left(x^2+y^2+z^2\right)^{9/2}} & -\frac{9\left(x^4+\left(2y^2-8z^2\right)x^2+y^4-8y^2z^2+\frac{8z^4}{3}\right)\mu J2 req^2}{2\left(x^2+y^2+z^2\right)^{9/2}} & 0\,0\,0
\end{bmatrix}
$$

Figure A.1: Jacobian Matrix of $J_2$ dynamics computed using Maple 2016.

$$
\begin{bmatrix}
-\frac{15\mu x J2 req^2\left(4x^4+x^2y^2-41x^2z^2-3y^4+15y^2z^2+18z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & -\frac{15 req^2 J2 y\mu\left(6x^4+5x^2y^2-51x^2z^2-y^4+5y^2z^2+6z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & -\frac{15 req^2 J2\mu z\left(18x^4+15x^2y^2-41x^2z^2-3y^4+y^2z^2+4z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & 0\,0\,0 \\
-\frac{15 req^2 J2 y\mu\left(6x^4+5x^2y^2-51x^2z^2-y^4+5y^2z^2+6z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{15\mu x J2 req^2\left(x^4-5x^2y^2-5x^2z^2-6y^4+51y^2z^2-6z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & -\frac{315\mu y J2 req^2 zx\left(x^2+y^2-2z^2\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & 0\,0\,0 \\
-\frac{15 req^2 J2\mu z\left(18x^4+15x^2y^2-41x^2z^2-3y^4+y^2z^2+4z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & -\frac{315\mu y J2 req^2 zx\left(x^2+y^2-2z^2\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{45\mu x J2 req^2\left(x^4+2x^2y^2-12x^2z^2+y^4-12y^2z^2+8z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & 0\,0\,0 \\
0 & 0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0\,0\,0
\end{bmatrix}
$$

Figure A.2: Hessian Matrix of the velocity x-component of $J_2$ dynamics computed using Maple 2016.

$$
\begin{bmatrix}
-\frac{15 req^2 J2 y\mu\left(6x^4+5x^2y^2-51x^2z^2-y^4+5y^2z^2+6z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{15\mu x J2 req^2\left(x^4-5x^2y^2-5x^2z^2-6y^4+51y^2z^2-6z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & -\frac{315\mu y J2 req^2 zx\left(x^2+y^2-2z^2\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & 0\,0\,0 \\
\frac{15\mu x J2 req^2\left(x^4-5x^2y^2-5x^2z^2-6y^4+51y^2z^2-6z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{45\mu\left(-\frac{4y^4}{3}+\left(-\frac{x^2}{3}+\frac{41z^2}{3}\right)y^2+x^4-5x^2z^2-6z^4\right)req^2 J2y}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{45\mu z\left(-\frac{4z^4}{3}+\left(-\frac{x^2}{3}+\frac{41y^2}{3}\right)z^2+x^4-5x^2y^2-6y^4\right)req^2 J2}{2\left(x^2+y^2+z^2\right)^{11/2}} & 0\,0\,0 \\
-\frac{315\mu y J2 req^2 zx\left(x^2+y^2-2z^2\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{45\mu z\left(-\frac{4z^4}{3}+\left(-\frac{x^2}{3}+\frac{41y^2}{3}\right)z^2+x^4-5x^2y^2-6y^4\right)req^2 J2}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{45\mu J2 req^2 y\left(x^4+2x^2y^2-12x^2z^2+y^4-12y^2z^2+8z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & 0\,0\,0 \\
0 & 0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0\,0\,0 \\
0 & 0 & 0 & 0\,0\,0
\end{bmatrix}
$$

Figure A.3: Hessian Matrix of the velocity y-component of $J_2$ dynamics computed using Maple 2016.

$$\begin{bmatrix} -\frac{15\,req^2\,J2\,\mu\,z\left(18x^4+15x^2y^2-41x^2z^2-3y^4+y^2z^2+4z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & -\frac{315\,\mu\,y\,J2\,req^2\,zx\left(x^2+y^2-2z^2\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{45\,\mu\,x\,J2\,req^2\left(x^4+2x^2y^2-12x^2z^2+y^4-12y^2z^2+8z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & 0\;0\;0 \\ -\frac{315\,\mu\,y\,J2\,req^2\,zx\left(x^2+y^2-2z^2\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{45\,\mu\,z\left(-\frac{4z^4}{3}+\left(-\frac{x^2}{3}+\frac{41y^2}{3}\right)z^2+x^4-5x^2y^2-6y^4\right)req^2\,J2}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{45\,\mu\,J2\,req^2\,y\left(x^4+2x^2y^2-12x^2z^2+y^4-12y^2z^2+8z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & 0\;0\;0 \\ \frac{45\,\mu\,x\,J2\,req^2\left(x^4+2x^2y^2-12x^2z^2+y^4-12y^2z^2+8z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{45\,\mu\,J2\,req^2\,y\left(x^4+2x^2y^2-12x^2z^2+y^4-12y^2z^2+8z^4\right)}{2\left(x^2+y^2+z^2\right)^{11/2}} & \frac{225\,\mu\,z\left(\frac{8z^4}{15}+\left(-\frac{8x^2}{3}-\frac{8y^2}{3}\right)z^2+\left(x^2+y^2\right)^2\right)req^2\,J2}{2\left(x^2+y^2+z^2\right)^{11/2}} & 0\;0\;0 \\ 0 & 0 & 0 & 0\;0\;0 \\ 0 & 0 & 0 & 0\;0\;0 \\ 0 & 0 & 0 & 0\;0\;0 \end{bmatrix}$$

Figure A.4: Hessian Matrix of the velocity z-component of $J_2$ dynamics computed using Maple 2016.