POLITECNICO
MILANO 1863

Department of Aerospace Science and Technology
Master's Thesis in Space Engineering

Application of hybrid differential dynamic programming in orbital elements for low thrust trajectory optimisation

Simone Carminati

Advisor:
Prof. Camilla Colombo
Co-Advisor:
Dr. Marco Nugnes

Academic Year 2021-2022, May 2023

This content is original, written by the Author, Simone Carminati. All the non-originals information, taken from previous works, are specified and recorded in the Bibliography.

When referring to this work, full bibliographic details must be given, i.e.

Carminati Simone, "Application of hybrid differential dynamic proggraming in orbital elements for low thrust trajectory optimisation". 2023, Politecnico di Milano, Master Thesis in Space Engineering, Supervisor: Camilla Colombo, Co-supervisor: Marco Nugnes.

Abstract

In the field of orbital mechanics, from its earliest beginnings, the optimisation of trajectories has always been a key step in deciding the feasibility of a space mission and its development. In recent years, modern technological advances have led to the introduction of electric thrusters, which give rise to new scenarios and mission possibilities that were hitherto completely precluded. In fact, they provide considerable benefits in terms of consumption compared to classical chemical thrusters, at the price of much more limited thrust capacities and, consequently, significantly longer transfer times. For these reasons, there is currently a strong push in the direction of using such engines in a large variety of situations, such as interplanetary travel, planetary orbital transfers, manoeuvring satellites belonging to constellations, low-thrust deorbiting strategies and even innovative concepts such as formation flying and in-orbit servicing.

The field of numerical optimisation was also influenced by this innovation and had to adapt to keep up. Indeed, the impulsive manoeuvre approximation typically used for chemical thrusters loses its validity in this case, due to the considerably longer burn time associated with modern electric thrusters. At this point the problem addressed, in order to produce plausible results, requires a continuous-thrust model, which assumes a large number of decision variables. Therefore, the transfer trajectories are now obtained as a solutions of a large scale highly nonlinear optimal control problem. Several optimisation methods have been developed to solve the latter, typically divided into two macro-families: direct methods and indirect methods. Of these, Differential Dynamic Programming (DDP) assumes particular importance, because it presents the advantages of methods from both families (accuracy on the one hand, robustness on the other) and a low increase in computational cost as the number of decision instants increases. It, therefore, turns out to be a powerful optimisation tool, which is particularly suitable for solving problems of large dimensions just as it is in the case of low-thrust trajectories. Despite its potential, until now this approach has been used in a fairly limited manner. Most of the studies that dealt with it for celestial mechanics problems in any case exploited the use of classical Cartesian coordinates as state variables.

In this thesis we adapt the latest well-validated version of this optimisation method, called

Hybrid Differential Dynamic Programming (HDDP), to a new set of variables often used in celestial mechanics, namely the orbital elements. This set reformulates the problem in terms of size, shape and orientation in the space of the orbit in which it lies, and has numerical advantages. In fact, their evolution in time during a generic transfer is known to be more regular and smooth than the oscillatory behaviour of the previous Cartesian coordinates. Thanks to this characteristic, it is expected that the method is facilitated in the optimisation and it converges faster to the solution. Furthermore, for some specific low-thrust applications that are particularly sensitive to the oscillation of the state variables and the variation of the control law, as it is the case of multi-revolution transfers in a planetary environment, the use of such variables is expected not only to improve the performance of the algorithm in terms of convergence speed, but also to extend its applicability to a number of revolutions previously impossible to handle.
In conclusion, this thesis work aims to assess the veracity of the aforementioned hypothesis by applying the optimisation method with the new set of variables to various practically relevant situations in the context of low-thrust trajectory design, through interplanetary and planetary transfers, in single and multi-revolution cases.

Keywords: Differential Dynamic Programming, Orbital elements, Trajectory Optimisation, Low-Thrust Trajectory

Sommario

Nell'ambito della Meccanica Orbitale, fin dalle sue origini, l'ottimizzazione di traiettorie è sempre stata un passo fondamentale per decretare la fattibilità di una missione spaziale e il suo successivo sviluppo. Negli ultimi anni, i recenti progressi tecnologici hanno portato all'introduzione dei propulsori elettrici, i quali danno vita a nuovi scenari e possibilità di missione finora del tutto precluse. Infatti, essi offrono notevoli vantaggi in termini di consumo rispetto ai classici propulsori chimici, al prezzo di livelli di spinta molto più limitati e, di conseguenza, di tempi di trasferimento significativamente più lunghi. Per questi motivi, attualmente c'è una forte spinta del settore per estendere l'utilizzo di tali motori ad una grande varietà di situazioni, come i viaggi interplanetari, i trasferimenti orbitali planetari, le manovre di satelliti appartenenti a costellazioni, strategie di deorbiting a bassa spinta e persino concetti innovativi come formation flying e in-orbit servicing.

Anche il campo dell'ottimizzazione numerica è stato influenzato da questa innovazione e si è dovuto adattare per tenere il passo. Infatti, l'approssimazione di manovra impulsiva tipicamente utilizzata per i propulsori chimici perde la sua validità in questo caso, a causa del tempo caratteristico considerevolmente più lungo associato ai moderni propulsori elettrici. A questo punto il problema affrontato, per produrre risultati plausibili, richiede un modello a spinta continua, che presuppone un elevato numero di variabili decisionali. Pertanto, le traiettorie di trasferimento a questo punto sono ottenute come soluzioni di un problema di controllo ottimo altamente non lineare di grande dimensione. Per risolvere quest'ultimo sono stati sviluppati diversi metodi di ottimizzazione, tipicamente divisi in due macro-famiglie: metodi diretti e metodi indiretti. Tra questi, la programmazione dinamica differenziale (DDP) assume particolare importanza, in quanto presenta i vantaggi dei metodi di entrambe le famiglie (accuratezza da un lato, robustezza dall'altro) e un basso incremento del costo computazionale all'aumentare del numero di istanti di decisione. Si rivela quindi un potente strumento di ottimizzazione, particolarmente adatto alla risoluzione di problemi di grandi dimensioni proprio come avviene nel caso delle traiettorie a bassa spinta. Nonostante il suo potenziale, finora questo approccio è stato utilizzato in modo piuttosto limitato. La maggior parte degli studi che lo vedono coinvolto per la trattazione di problemi di meccanica celeste ha in ogni caso adottato le classiche coordinate cartesiane come variabili di stato.

In questa tesi si punta ad adattare la versione più recente e ben validata di questo metodo di ottimizzazione, chiamato Programmazione Dinamica Differenziale Ibrida (HDDP), ad un nuovo set di variabili spesso utilizzate in meccanica celeste, ovvero i cosiddetti elementi orbitali. Questo set riformula il problema in termini di dimensione, forma e orientamento nello spazio dell'orbita in cui si trova, e presenta diversi vantaggi numerici. Infatti, è ben noto che l'evoluzione nel tempo di tali elementi durante un generico trasferimento è più regolare e più stabile del comportamento oscillatorio delle precedenti coordinate cartesiane. Grazie a questa caratteristica, ci si aspetta che il metodo sia facilitato nell'ottimizzazione e che converga più velocemente alla soluzione. Inoltre, per alcune applicazioni specifiche a bassa spinta che risultano particolarmente sensibili all'oscillazione delle variabili di stato e alla variazione della legge di controllo, come nel caso dei trasferimenti multirivoluzione in ambiente planetario, si ipotizza che l'uso di tali variabili non solo migliori le prestazioni dell'algoritmo in termini di velocità di convergenza, ma che consenta anche di estendere la sua applicabilità a un numero di rivoluzioni finora impossibile da gestire.
In conclusione, il presente lavoro di tesi si pone come obiettivo quello di valutare la veridicità delle ipotesi sopracitate applicando il metodo di ottimizzazione con il nuovo set di variabili a diverse situazioni di interesse pratico nel contesto della progettazione di traiettorie a bassa spinta, attraverso trasferimenti sia interplanetari che planetari, in entrambi i casi di rivoluzione singola e multipla.

Parole Chiave: Programmazione Dinamica Differenziale, Elementi Orbitali, Ottimizzazione di Traiettorie, Traiettorie a Bassa Spinta

Contents

Nomenclature

DDP Differential Dynamic Programming

ECI Earth-Centered Inertial

GEO Geosynchronous Equatorial Orbit,commonly known as Geostationary Orbit

GTO Geostationary Transfer Orbit

HBVP Hamiltonian Boundary Value Problem

HDDP Hybrid Differential Dynamic Programming

ICRF International Celestial Reference Frame

ISRO Indian Space Research Organization

LVM 3 Launch Vehicle Mark 3

NLP Non Linear Programming

RTG Radioisotope Thermoelectric Generator

STM State Transition Matrix

TOF Time Of Flight

TRQP Trust Region Quadratic sub-Problem

1 | Introduction

1.1 Background

In recent years, interest in low-thrust trajectories for orbital transfers has been rapidly growing. This topic attracts the attention of many scientists because of the considerable savings in propellant mass that today's electric thrusters allow due to their high specific impulses. This aspect allows a great increment of the payload capacity associated with a spacecraft, opening the door to new mission's possibilities that were hitherto completely precluded.
Another typical feature of these engines is a low deliverable thrust with respect the nominal one that a chemical thruster can achieve. In this context electric engines are particularly suitable for applications where the forces involved are small and time is not a primary concern. In the field of Orbital Mechanics it can be, for example, the case of an interplanetary leg performed in deep space or the orbit maintenance of a satellite around a celestial body. However, due to recent improvements in space sector, more and more frequently electric engines are also designated as primary source to perform orbital transfer in planetary environment.
The design and the optimisation of these kind of trajectories is a problem that pose some challenges with respect the design of a transfer characterised by impulsive manoeuvres, which is the general approximation done whenever a chemical thruster is exploited. Indeed, for an electrical engine, the characteristic time of the manoeuvre is longer due to the low-level of thrust. This is the reason why a continuous-thrust model shall be employed, which implies a large number of control/decision variables that are involved in the optimisation problem.
Moreover, especially for the planetary case, a transfer between two orbits in general require a lot of revolutions. This usually introduces characteristic time and distances very different inside the same problem, which makes the numerical optimisation process more difficult to be solved.

The choice of a particular set of variable assume, in this context, a fundamental importance. Indeed, especially for multi-revolution case, Cartesian variables has an oscillatory behaviour that requires a fine discretisation in order to don't prevent the convergence of the optimisation algorithm. Whereas, exploiting a set of variables that has a smoother evolution along the transfer, can reduce a lot the number of nodes required to reach convergence. This is due to optimisation method and also due to the accuracy achieved by the integration of the trajectory during process. In most of the cases, and especially in differential dynamic programming, the two aspects are not so independent of each other.

1.2 Aim of the thesis

For these reasons presented in section 1.1, the aim of the work here presented is to show how it may be useful to couple a robust optimisation method, a revised version of Hybrid Differential Dynamic Programming (HDDP) [27], with a coherent set of state variables that limits the numerical instabilities, the orbital elements [21]. This is something which has been done only by [23] and [32] before, and due to the poorness of the literature that regards this topic is useful to investigate about this coupling further for the benefit that it brings.
This dissertation's purpose is to offer a solution for solving problems involving low-thrust and continuous-thrust trajectories in an efficient and safe manner, emphasising the limits and the necessary precautions of the method through the analysis of some practical applications.

1.3 Novel contributions

The major novel contribution is related to the coupling of the HDDP optimisation method to work in orbital elements. Indeed, as will be presented later, the selection of this new set of variables brings several advantages from an optimisation viewpoint, but also some drawbacks that shall be considered. Some precautions are indeed required on the dynamic definition and set selection to grant the success of an optimisation done by HDDP. In this thesis, it will be analysed in detail all the aspects related to this coupling and its flexibility, by the application of the method to four very different situations relevant in orbital mechanics.

1.4 Thesis structure

The master thesis will be structured in the following way. As first, an overview about the optimal control problem and different optimisation method will be given in order to

introduce the reader in the context in which this work aims to fit. Then, a more detail about the differential dynamic programming and its most recent improvements will be analysed. at this point, the major contribution of this work are reported, focusing on the coupling of HDDP and orbital elements. Subsequently, by means of four different practical applications, all aspects of the optimisation method will be analysed. Finally, the work will be concluded by summarising the results obtained and possible future developments.

2 | Literature review

2.1 Optimal Control Problem

In this section an overview on what is an Optimal Control Problem is provided for sake of clarity and to frame the problem addressed in this work. The next section will then go on to describe the solving methods in order to clarify the state of the art regarding optimisation algorithms.

An optimal control problem consists in the minimisation of an objective function J, that depends on a control law u and its associated trajectory x. The procedure typically consists in iterative methods that progressively change u and x until the so-called "optimal path" is achieved, i.e. the path and the control law that minimise J.
The trajectory usually is generated by a dynamic that shall be respected during the optimisation. Moreover, sometimes other additional constraints can be present (for example the final point that the path shall match, the maximum values of the control laws, and so on).

Its solution consists in inserting the constraints inside the cost-function trough the use of appropriate multipliers and then imposing that the derivatives of J with respect to all the different variables (multipliers included) are equal to 0. This procedure leads to the formulation of the so-called Hamiltonian function and the Euler-Lagrange equations, whose resolution leads to the minimum sought.

2.2 Optimisation methods

The solving methods of an optimal control problems belongs to two main families: Direct methods and Indirect methods.

Indirect methods are based on the calculus of variations [1] and Pontryagin's Maximum Principle [26] in order to find the stationary point associated to the solution. The first step is to write the Hamiltonian function and than, imposing its derivatives equal to zero, the Euler-Lagrange differential equations are retrieved. Solving this set of equations, which means solving an Hamiltonian Boundary-Value Problem (HBVP) [4], leads to a solutions which is intrinsically optimal. This approach however, introduces in the problem additional variable to determined, the so-called "co-states", which are the factors through which the constraints are incorporated into the Hamiltonian. The physical meaning of these variables is not always perfectly clear at priori, but they take parts in the solving process of the problem and directly influence it.
Usually the set of differential equations is solved numerically, and to do that generally an initial guess of the state variables shall be provided to start the solving process, adjoint states included; moreover, for a lot of problems faced with this approach, the convergence region is quite strict.
This aspects leads to these two big difficulties:

- Being not easy to define the physical interpretation of the co-states, defining their first attempt value also becomes a difficult step.
- The selection of the first attempt value of the co-states often results outside the attraction's region and this prevents the convergence of the solution.

In addition, it should be also considered that the optimality differential equations associated to the adjoint variables are not so easy to be derived, especially if the dynamic of the problem is already complex in itself. Last but not least, also the integration of inequalities stage constraints results challenging, due to the fact that in general it should be defined at priori with respect the solving process if they are active or not.
To sum up, indirect methods present several difficulties that can only be solved if there is a deep physical knowledge of the problem. However, when there is the possibility to be able to do so, these approaches leads to solutions that are very accurate and which need a limited amount of nodes to reach convergence.

Direct methods on the other hand, somehow represent the other side of the coin. They consists in a reformulation of the problem, where the trajectory is firstly discretised and parameterised, and then the solution is searched exploiting a Non-Linear Programming (NLP) technique. The main idea is convert the differential problem into a parametric one, and then change the parameterised decision-vector obtained at each iteration obtaining results that better and better approximate the optimal solution sought. NLP methods typically rely on physical quantities and does not require cumbersome co-states to be defined. These aspects eliminate all the major difficulties encountered in indirect methods and increase a lot robustness, allowing to achieve convergence also with initial guesses which are quite far

from the optimal solution. On the other hand, due to the approximation introduced by the parameterisation, the final results founded are quite inaccurate when compared with the ones founded by indirect methods. Indeed in general, to achieve good level of accuracy, these methods require a large number of nodes.

Having explained the differences between the two main approaches, the major optimisation methods are now presented one by one. In general, each method can be generated by direct or indirect formulation, with their respective pros and cons.

The first methods presented is the single-shooting [5]. It consists in the propagation of the trajectory once defined the optimisation variables as initial condition, hence the shooting name. At the end of this process, the variables are modified as a function of a final error, in order to optimise trajectory and to match the constraints on the final state. A typical feature of this method is the big sensitivity of the problem to the initial variables. Indeed, due to the propagation of the integration of the whole trajectory, sometimes also small variations of optimisation parameters associated to the initial part of the trajectory can generate big variations at the end of it. This phenomenon manifest itself in a particularly pronounced manner whenever the dynamic is highly non-linear and produces some difficulties in the numerical optimisation process.

The multiple-shooting technique [40] attempts to solve this issue by subdividing the trajectory in different sub-intervals which are independently integrated one by the other. This feature allows to reduce the effect that a small variation on variables that affect first stages of the trajectory has at the end of it. However, the system to be solved at each iteration to handle the constraints is much larger with respect to the single shooting one. This is due to additive constraints that shall be included in the formulation between two consecutive leg to grant the continuity of the trajectory. This aspect increase the computational effort of the method. Moreover, for both multiple and single shooting, the integration of stage-constraints in the problem add another difficulty. Indeed, in this case is necessary to subdivide between constrained and unconstrained arcs of the trajectory even before to start the process. In practical words this means knowing at priori when a constraint is active or not along the path, which is almost impossible in most of the cases.

Collocation methods [22] take another important role in the optimization techniques. They consists in exploiting piece-wise (in general) polynomial interpolation to approximate the trajectory. Differently from the shooting technique, collocation does not require numerical integration of the trajectory, but the dynamic differential equations are enforced as a constraint at some specific points. These latter lie in each stage interval in which the path is subdivided, and the dynamic is included trough an algebraic process. From one side this aspect allows to reduce the computational effort per-stage due to the lack of a numerical integration. On the other hand, for the same reason, in order to generate a path that reasonably approximates the true dynamic of the problem, is necessary to have a quite large number of nodes for the discretisation, which increase the computational effort. An

important aspect of the direct collocation case is that it is possible to include in a more proper way some stage-constraints, applying technique of active set [36] or interior-point [43].

As can be seen there are many methods of optimisation that exploit different principles in order to obtain a solution. In relation to the Orbital Mechanics problems, both direct and indirect methods has been employed. Regarding indirect ones some advances has been made thanks to the works of Edelbaum [41] [42], Wiesel and Alfano [44], Petropulos [3] [2] and Kluever [10]. On the other hand, examples of direct approach are reported in the work of Betts [8] [9] [24].
However each method has its own drawbacks. The major ones of all the indirect approaches consist in the restrictions imposed by the manage of the adjoint states, which limit the complexity of the problem addressed and require the knowledge of it in order to reach convergence. For what concern all the direct approaches, due to the approximation introduced by discretisation, a lot of nodes in which subdivide the path are needed to achieve accurate results. At the same time, in general the NLP technique exploited to optimise the variables deals with the inversion of big matrices that increase rapidly in dimensions with the number of discretisation's nodes. Therefore, the direct methods results computationally heavy, limiting practically the dimension of the problem that they can face.

2.3 Differential Dynamic Programming developments

Differential Dynamic Programming assume, in the context just presented, an important relevance. It is a cross between a direct and an indirect method, and it perfectly fits the need requested by the solving process of large-scale problems. Indeed it is generally classified as a direct method, because it does not introduce any adjoint state in the formulation. However, differently from the other direct methods, DDP has a strict connection with the calculus of variations. In fact, it can be shown (as demonstrated by Dreyfus [39]), that the DDP procedure minimises the Hamiltonian of the problem, as in the calculus of variations. In this way DDP does not formulates the problem by explicitly exploiting the optimality conditions as indirect methods do, but the solution found is nevertheless influenced by them. DDP therefore inherits the robustness typical of the direct methods and accuracy similar of the indirect ones. Moreover this technique presents quadratic convergence under certain hypothesis and, as it will be explained better in the next section, is more computationally efficient if compared with the classical direct methods. All the aforementioned aspects contribute in fuelling interests towards DDP, which is increasingly regarded as an optimisation approach with interesting possible future developments and has already become the state of the art for robust complex trajectory design.

Differential Dynamic Programming was introduced for the first time by Mayne [15], which develops a first basic version of the algorithm suitable for unconstrained discrete-time problems. Then other improvements are provided by Gershwin and Jacobson [18], Jacobson and Mayne [25], Dyer and Mc Reynolds [16], Yakowitz and Murray [33]. Indeed, from its first formulation, the potential of DDP was immediately realised and many authors start to give their contribute with some improvements in order to expand its applicability in an ever-widening manner.
One of main concern was to try to include in an efficient way the capability of the algorithm to face constrained problem. The other important focus was related to the possibility of dealing optimally even with problems featured by highly non-linear dynamics, where the Hessian can occasionally becomes no longer positive-definite (this difficulties will be better explained in the next section, but it is a fundamental aspect to grant a minimisation when the algorithm is running).
Regarding the first aspect, considerable progress was made; most of them consisted in the application of an augmented cost function, in which, through the use of appropriate multipliers, constraints also appear. On the other hand, concerning the second, the problem was solved introducing a "shifting" of the Hessian. One of the first technique of this kind was suggested by Liao and Shoemaker [29] [30], but also other methods are exploited, like the one presented in Colombo et al.[13].
At the moment, the current state-of-the-art technology for the optimal design of low-thrust trajectories that exploits a differential dynamic programming approach is the Mystic Low-Thrust Trajectory Design and Visualization Software [34]. Mystic has best demonstrated its capabilities with the success of NASA's Dawn mission [11], and its optimisation engine is built around the Static/Dynamic Optimal Control algorithm, a differential dynamic programming approach developed by Whiffen [17].
More recently, the most major contribution to the theoretical field of DDP is provided by the work of Lantoine and Russel [27], which developed a new version of the algorithm called Hybrid Differential Dynamic Programming (HDDP). It mixes a Differential Dynamic Programming approach with sophisticated mathematical techniques in order to increase the robustness, computational efficiency and applicability. The paper is of paramount importance for this dissertation and will therefore be presented more in detail in the next chapters.
From the work of Lantoine and Russel [27] other developments has been introduced in order to increase computational efficiency, generally trying to reduce the most computationally expensive steps of the algorithm, and also in order to broaden the applicability to very sensitive problems, like multi-revolution planetocentric transfers. Regarding the first purpose, in general the focus is devoted in reducing the time needed to compute the state transition matrices of the trajectory. Maestrini [31] proposed a solution exploiting differential algebra to approximate the STM in a efficient way without the need of integration. Russel and Pellegrini [37] [38] instead propose a solution that exploits parallelisation to its full potential, where HDDP is reformulated in a multiple-shooting approach that allows not only the parallelisation of the STMs' integration but also the entire DDP process associated with

each phase of the trajectory. Concerning the improvements to adapt DDP to problems featured by many revolutions, a significant contribution is given by Aziz [23] and Colombo and Nugnes [12]. Aziz [23] introduces a transformation to change the independent variable from time to true/eccentric anomaly, whereas Colombo and Nugnes [12] face the problem exploiting orbital elements. Both of these works, although in different ways, allow to increase a lot the number of revolutions that can be achieved by the solution of the problem faced.

3 | Differential Dynamic Programming

In this chapter, the basic DDP algorithm and its recent improvements are presented in order to grant a better understanding of the work. First, the reader is introduced to the mathematical notation adopted during the whole dissertation and to the problem formulation, then the fundamental steps of the basic version of the DDP algorithm are described, before finally moving on to the last largely validated upgrade of the algorithm, which consists in Hybrid Differential Dynamic Programming (HDDP).

3.1 Mathematical notation

Due to the presence of a lot parameters and derivatives in the discussion, in this section an attempt to remove any ambiguity of notation is performed. Indeed, for a general multi-stage multi-phase problem, some confusion can arise due to presence of multiple subscripts that describe the node related to the *j-th* stage at the *i-th* phase. The whole trajectory of a problem, in fact, can be divided in M phases, each one is divided in a number N_i of stages. A phase represents a "leg" of the trajectory. In a problem of orbital mechanics related to an interplanetary travel, per example, it can be associated to the fraction of trajectory in which a spacecraft is still inside the sphere of influence of a planet, or the fraction of trajectory in which the spacecraft moves into the deep space, far by other celestial bodies. Each phases can have its own dynamics. The stages instead, represent the nodes in which each phase is discretised. The number of stages is often responsible of the accuracy achieved by the solution, due to a more/less refined numerical integration of the trajectory and also the big/small number of degrees of freedom associated to the control policy.

In this context it is important to define what the subscripts refers to and to what each quantity represents in the description.
In the work presented, only single-phase problems are faced, so this ambiguity is avoided.

However, it is important for sake of clarity, to define in an unequivocally manner all the mathematical operators/entities that will appear afterwards in the dissertation.
Because the algorithm exploited for this work is strictly related to HDDP version developed by Lantoine and Russel [27], all the symbols reported in the following list follows the same notation, referring to single-phase problems. For a complete description that involves also multi-phase problems the reader can refer to [27].

$x_k \in \mathbb{R}^n$ are the states of dimension n at stage k

$u_k \in \mathbb{R}^m$ are the dynamic controls of dimension m at stage k

$w \in \mathbb{R}^p$ are the static parameters of dimension p

$f_k \in \mathbb{R}^n \to \mathbb{R}^n$ are the dynamic functions that represents the first derivatives of the states as a function of state, control inputs and static parameters at each stage

$F_k \in \mathbb{R}^n \to \mathbb{R}^n$ are the transitions functions that propagates the states across each stage

$L_k \in \mathbb{R}^n \to \mathbb{R}$ are the stage cost functions

$\phi \in \mathbb{R}^n \to \mathbb{R}$ are the final state cost function

$g_k \in \mathbb{R}^n \to \mathbb{R}^{n_g}$ are the stage constraints

$\psi \in \mathbb{R}^n \to \mathbb{R}^{n_\psi}$ are the final state constraints

$J \in \mathbb{R}$ is the objective function of the problem to be minimised

$J^* \in \mathbb{R}$ is control-free cost where control is replaced by the state dependent control law: $J^*(x) = J(x, u(x))$

$J_k \in \mathbb{R}$ is the Cost-to-go function at the stage k

$J_{q,k} \in \mathbb{R}^q$ is the first partial derivative of J_k with respect to the dummy vector variable q at stage k (column-wise convention)

$J_{qq,k} \in \mathbb{R}^q \times \mathbb{R}^q$ is the second partial derivative of J_k with respect to the dummy vector variable q at stage k

The last aspect to be emphasised is the presence of tensor operations in the formulation. Indeed, it is important to clarify how these operations are performed to avoid ambiguity or confusion in the definition of different operators.
The operations that involves third-order tensor in this work are expressed by means of $\bullet$ operator. Defined B as a tensor, we can have:

- $C = A \bullet B \implies C(i,j) = \sum_{p=1}^{N} A(p)B(p,i,j)$, when A is a vector (C is a matrix)
- $C = A \bullet B \implies C(i,j,k) = \sum_{p=1}^{N} A(i,p)B(p,j,k)$, when A is a matrix (C is a third-order tensor)

3.2 Problem formulation

A dynamical system is described by the differential equation:

$$\dot{x} = f(x(t), u(t), w; t) \tag{3.1}$$

where $x = [x_1, ..., x_n]^T$ is a vector of n variables that represent the state vector, $u = [u_1, ..., u_m]^T$ is a vector of dimension m that represent the control input, and $w = [w_1, ..., w_p]^T$ is a vector of dimension p that represent the so-called "static parameters", a number of variables that does not depend on time but that influence the dynamic behaviour of the system. Sometimes, in discrete-time, this formulation can be substituted in the following way, thanks to the introduction of the transition function F_k:

$$x_{k+1} = F_k(x_k, u_k, w) \tag{3.2}$$

where the link between F_k and f_k is represented by:

$$F_k = x_k + \int_{t_k}^{t_{k+1}} f_k(x, u_k, t)dt \tag{3.3}$$

Given a dynamical system described by the aforementioned equations, solving an optimal

control problem consists, once fixed the initial state x_0, in finding the control law $u(t)$ and the static parameters w such that the cost-function J is minimised:

$$J := \phi(x_{t_f}) + \int_{t_0}^{t_f} L(x(t), u(t))dt$$

where ϕ is a function that depends only by the final state, whereas L is a merit function that is dependant on all the remaining stages and respective controls of the trajectory. In discrete-time the equation becomes:

$$J := \sum_{j=0}^{N} L_k(x_k, u_k) + \phi(x_{N+1}) \tag{3.4}$$

Sometimes there may also be some constraints that shall be respected: stage-constraints, which express some constraints on each single stage of the trajectory and which have the following form:

$$g_k(x_k, u_k, w) \leq 0 \tag{3.5}$$

and final state constraints, which in general represents the desired final state that the trajectory shall reach and typically have the following equality form:

$$\psi(x_{N+1}, w) = 0 \tag{3.6}$$

3.3 Differential dynamic programming

Differential dynamic programming is a modern way to face an optimal control problem of large dimension. It consists of an iterative method that produces consecutive trajectories associated with a gradually improved cost value, until a local minimum is reached.
It is based on the philosophy of dynamic programming, which, through a backward process, produces a "family" of optimal paths leading to the same final point but with different initial conditions. This why the term "fields of extremals" has been coined in the literature [35]. At each step of the regression in fact, the new optimal path is formed by the one obtained at the previous step with the addition of a new node that represent a new initial condition. The path is made optimal, and then the process is repeated until the starting state of the trajectory is the initial condition of an optimal path. Considering the solution obtained, each portion that connects one of its generic points at the final one, is therefore an optimal path.
This backward methodology is based on Bellman's optimality principle [35], which generates a new optimal path via the following backward recursive equation:

$$J_k(x_k) := L_k(x_k, u_k) + J^*_{k+1}(x_{k+1}) \tag{3.7}$$

where J_k is defined as the cost-to-go function, a function that represent the cost value of the trajectory that goes from node k to the final one; J^*_{k+1} is the optimal value of the cost-to-go function associated to the optimal path that goes from node $k+1$ to the final one; L_k is a the stage cost function.

This is an equivalent reformulation of the Hamilton-Jacobi optimality principle, but its application is obtained in different way with respect to the classical variational approach, as already explained.
Dynamic Programming approach grants the achievement of a global minimum solution in just one iteration. However it has an insurmountable drawback: the computational cost. Indeed, in order satisfy the Bellman equation it is necessary to discretize both state variables and time. Then, an optimal cost value shall be associated to each discrete point x_k of the grid generated at each time step k. All these values shall be stored to pass to the next time step and proceed with the backward process. This procedure implies therefore a huge amount of data to be stored; amount that, being already enormous, increase exponentially with the dimension of state and control. It means that, for a problem with 6 state variables and 3 control inputs (typical dimensions for an orbital mechanics problem), although theoretically the solution is obtained in a single iteration, the computational effort is so prohibitive as to prevent its effective application in practice. This problem, according to Bellman's denomination, is called *"curse of dimensionality"*.

Differential Dynamic Programming fixes this issue, but the solution obtained is only a local minimum. Indeed, with this approach, the Bellman equation is approximated with a second order expansion around a reference trajectory and a reference control policy. At this point the data to be stored in order to pass from the *(k+1)-stage* to the *k-stage* during the backward process are only the coefficients of the Taylor expansion. In this way the computational effort is extremely reduced, making the process applicable, but the method requires several iterations to reach the optimal solution. The minimum found, moreover, lose its globality property and reduces only to a local one.

This method is more computationally-efficient with respect to the other classical direct methods as previously anticipated. Indeed, considering the solving process at each iteration of a problem with n state variables, m control inputs and N discretization stages in time, a typical NLP technique consists in solving a single large linear system, which requires the inversion of matrices of order $Nm \times Nm = O(N^2)$. It means that the computational cost grows rapidly as the number of discretization stages increases. DDP instead, breaks the initial problem in N smaller problems to be solved, where at each step, the only variable to be optimized is the control input of the current time step k. Therefore the matrices that the method have to deal with are N and have always dimensions equal to $m \times m$. This feature implies an increment of the computational effort directly proportional to the number

of the discretization stages $O(N)$, making the method particularly suitable for large-scale problems.

DDP is an approach that is composed of two main procedures: Backward Induction and Forward Propagation. During the first, from a reference control law and its associated trajectory, an optimal control variation and the expected reduction of the cost-function is computed. Whereas in the second, the new trajectory in propagated, paying particular attention that it falls in the neighborhood of the reference one, without violating the limits imposed by the approximation of the second order expansion.
In the following sections an overview on the most important steps of the aforementioned phases of the DDP is provided. For sake of clarity, for the moment static controls and constraints are ignored.

3.3.1 Backward induction

The aim of this phase is to select a proper optimal control law knowing a reference one and its associated trajectory, in order to minimize the objective function. The process consists in solving an optimal control problem at each time-step k trough the equation 3.7, where the unique variable to optimize is the control input u_k at that precise time instant. The idea behind this procedure is that, having previously computed an optimal path from the node $(k+1)$ to the final one, the path from the node k to the final one, in order to be optimal, will contain it. Doing so, the "last" part of the trajectory is already selected and does not influence the optimal control problem at the instant k, which instead can be influenced by the "new" part of the trajectory, i.e. the node k and its control input u_k. As Bellman itself reports in [35]:

> *"An optimal policy has the property that whatever the initial state and initial decision are, the remaining decision must constitute an optimal policy with regard to the state resulting from the first decision"*

To do so, the terms present in the Equation 3.7 are expanded up to the second order with respect to the variables at stake, around the reference trajectory $\bar{x}$ and the reference control $\bar{u}$.

$$J_k(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) = L_k(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) + J^*_{k+1}(\bar{x}_k + \delta x_k) \tag{3.8}$$

Expanding each term of the equation, the following expansion are obtained:

$$\begin{aligned} L_k(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) \approx L_k + L_{x,k}\delta x_k + L_{u,k}\delta u_k + \frac{1}{2}\delta x_k^T L_{xx,k}\delta x_k + \\ \frac{1}{2}\delta u_k^T L_{uu,k}\delta u_k + \delta x_k^T L_{xu,k}\delta u_k \end{aligned} \tag{3.9}$$

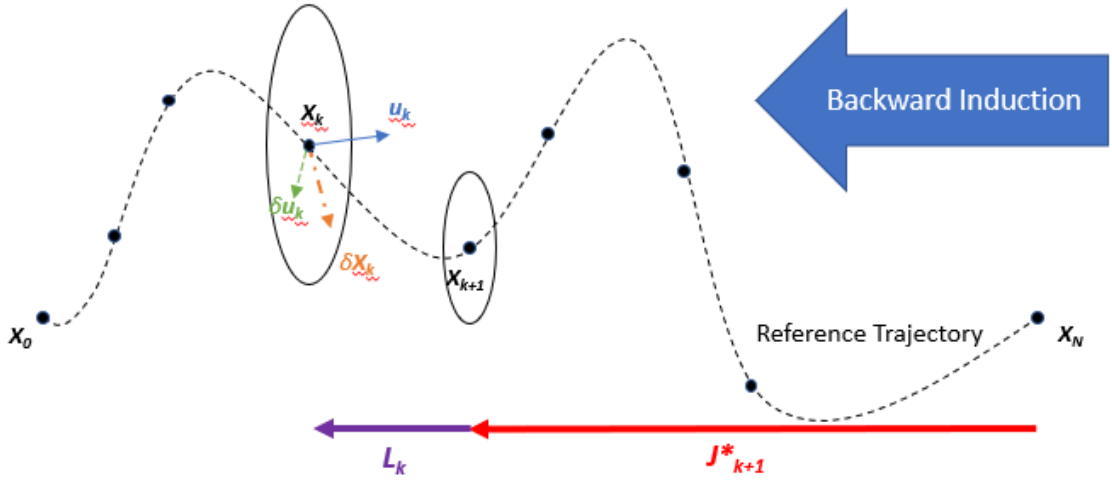


Figure 3.1: Conceptual Representation of the Backward Induction Process, image adapted from [31]

$$J_k(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) \approx J_k + J_{x,k}\delta x_k + J_{u,k}\delta u_k + \frac{1}{2}\delta x_k^T J_{xx,k}\delta x_k + \frac{1}{2}\delta u_k^T J_{uu,k}\delta u_k + \delta x_k^T J_{xu,k}\delta u_k \tag{3.10}$$

$$\delta J^*_{k+1} = J^*_{k+1}(\bar{x}_{k+1} + \delta x_{k+1}) \approx J^*_{k+1} + J^*_{x,k+1}\delta x_{k+1} + \frac{1}{2}\delta x_{k+1}^T J^*_{xx,k+1}\delta x_{k+1} \tag{3.11}$$

The latter term is not a function of u_k because it is the optimal value of the cost-to-go function, which is already the result of a minimization around $\bar{u}_k$. Indeed, starting from the Equation 3.7, we can obtain:

$$\begin{aligned} J_k &= L_k(x_k, u_k) + J^*_{k+1}(x_{k+1}) = J_k(x_k, u_k, x_{k+1}) \\ J_k &= L_k(x_k, u_k) + J^*_{k+1}(F(x_k, u_k))) = J_k(x_k, u_k) \\ J^*_k &= \min_{u_k}(J_k) = \min_{u_k}(L_k(x_k, u_k) + J^*_{k+1}(F(x_k, u_k))) \\ J^*_k &= J^*_k(x_k) \end{aligned}$$

At this point the dynamic equation is exploited in order to rewrite the optimal J^*_{k+1} as a function of state and control of the k-stage. Indeed, it can be written:

$$\delta x_{k+1}(x_k, u_k) \approx F_{x,k}\delta x_k + F_{u,k}\delta u_k + \frac{1}{2}\delta x_k^T F_{xx,k}\delta x_k + \frac{1}{2}\delta u_k^T F_{uu,k}\delta u_k + \delta x_k^T F_{xu,k}\delta u_k \tag{3.12}$$

The previous expression is substituted into equation 3.11, and then all the expansion are inserted in Eq. 3.8. After some manipulations the different terms of the equation can be collected. The results of this operation bring up to the needed partials of J_k.

$$\begin{aligned} J_k = & L_k(x_k, u_k) + J^*_{k+1}(x_{k+1}) \approx L_k + J^*_{k+1} + \begin{bmatrix} L_{x,k} + J^*_{x,k+1} F_{x,k} \\ L_{u,k} + J^*_{x,k+1} F_{u,k} \end{bmatrix}^T \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} + \\ & + \frac{1}{2} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} \end{aligned} \tag{3.13}$$

where:

$$\begin{cases} Q_{xx} = L_{xx} + J^*_{x,k+1} \bullet F_{xx,k} + F^T_{x,k} J^*_{xx,k+1} F_{x,k} \\ Q_{xu} = L_{xu} + J^*_{x,k+1} \bullet F_{xu,k} + F^T_{x,k} J^*_{xx,k+1} F_{u,k} \\ Q_{ux} = Q^T_{xu} \\ Q_{uu} = L_{uu} + J^*_{x,k+1} \bullet F_{uu,k} + F^T_{u,k} J^*_{xx,k+1} F_{u,k} \end{cases} \tag{3.14}$$

Then, in order to obtain a an optimal control law, the condition that shall be imposed is that $\frac{d}{d(\delta u_k)} J_k = 0$, leading to:

$$L_{u,k} + J^*_{x,k+1} F_{u,k} + Q_{ux} \delta x_k + Q_{uu} \delta u_k = 0 \tag{3.15}$$

From this equation the control variation δu_k is retrieved, at the condition that the Hessian with respect to the control Q_{uu} is positive definite. This condition is fundamental because it makes the stationary point (found by imposing the derivative of the control equal to zero) a local minimum of the cost function for the quadratic-approximation of the cost-function. In other words, the control variation found is progressively minimizing the cost-function.

$$\delta u_k = A_k + B_k \delta x_k \qquad \begin{cases} A_k = -Q^{-1}_{uu}(L_{u,k} + J^*_{x,k+1} F_{u,k}) \\ B_k = -Q^{-1}_{uu} Q_{ux} \end{cases} \tag{3.16}$$

where the coefficients A_k and B_k shall be stored during the Backward Sweep. They in fact represent the coefficient of the new feedback control policy, which will be exploited during the Forward Propagation in order to define an improved trajectory. Once retrieved this coefficients, the expression of δu_k can be substituted inside the expansion of J_k in order to find the optimal value of the cost-to-go function J^*_k.
Performing this procedure, the coefficients of the expanded optimal value J^*_k are retrieved by Eq. 3.17. Moreover, with this operation, in the expression of J_k appear some terms that represent the expected reduction of the cost-to-go function passing from the $(k+1)$-stage to the k-stage which are due to the variation of the control law.

$$
\begin{aligned}
ER_k &= ER_{k+1} + J_{u,k}^T A_k + \frac{1}{2} A_k^T J_{uu,k} A_k \\
J_{x,k}^* &= J_{x,k} + J_{u,k}^T B_k + \frac{1}{2} A_k^T J_{uu,k} B_k + A_k^T J_{ux,k} \\
J_{xx,k}^* &= J_{xx,k} + \frac{1}{2} B_k^T J_{uu,k} B_k + J_{ux,k}^T B_k + B_k^T J_{ux,k}
\end{aligned}
\tag{3.17}
$$

These optimal values of the derivatives of J and the expected reduction become the starting point that allow to recursively repeat the process at the next stage.
The procedure is repeated up to the initial node of the trajectory. As an initial condition whenever the backward recursion process is initialised, it is necessary to derive the derivatives $J_{x,N+1}^*, J_{xx,N+1}^*$ according to the defined objective function, and to set $ER_{N+1} = 0$.

3.3.2 Forward propagation

Once the coefficients for all stages are stored in memory the Backward Induction is to be regarded as completed. Now the new trajectory and the new control law are forward propagated starting from the initial node and applying recursively the following formulas, imposing that the initial conditions are $\delta x_0 = 0, \delta u_0 = A_0$.

$$
\begin{cases}
u_k = \bar{u}_k + \delta u_k = \bar{u}_k + \epsilon A_k + B_k(x_k - \bar{x}_k) \\
x_{k+1} = F(x_k, u_k)
\end{cases}
\tag{3.18}
$$

where ϵ is a parameter which is initially set equal to 1, and then progressively decreased if needed. Once the whole trajectory and the control law associated are computed, also the new value of the associated objective function J_{new} is computed.
At this point there is the need to check whether the trajectory just calculated is "close enough" to the previous one, to be sure that the second-order expansion approximation is not violated. To perform this operation the value of J_{new} and J are compared and it is checked if the variation of the cost-function is similar to the value of the expected reduction ER_0 computed on the whole reference trajectory. If this is not the case in fact, for non-linear problems the minimization is not granted at priori, because the second order approximation around the reference trajectory could not represent in a proper way the non-linear behaviour of the cost-function in the region of the new solution computed. If such a case occurs, the value ϵ is halved and the process is repeated, generating a new control law which is more similar to the reference one, and accordingly, also a trajectory more similar to the reference one. At this point the is check is repeated and the process is replicated until the checking criterion is met. This method in literature is known as a line-search method. The general criterion to accept a new trajectory for unconstrained problems is the following:

$$
J_{new} - J \leq \frac{\epsilon}{2} ER_0 \tag{3.19}
$$

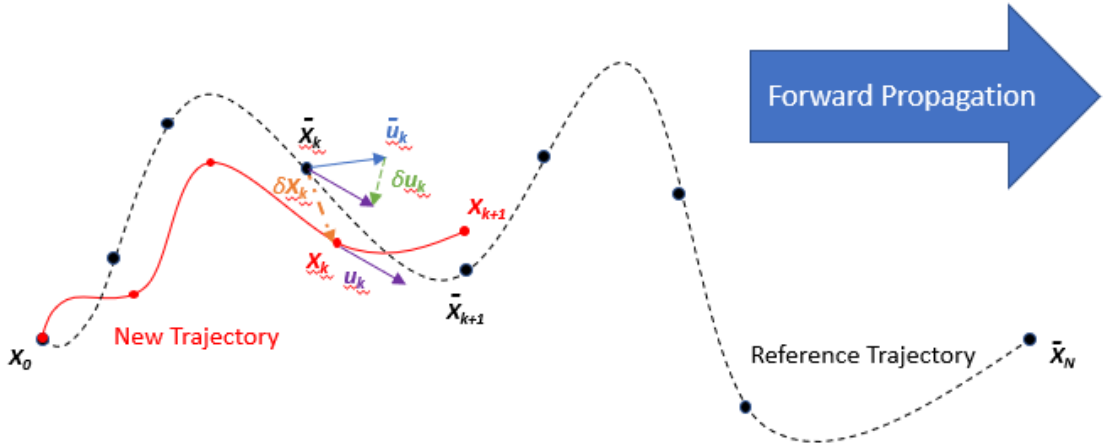


Figure 3.2: Conceptual Representation of the Forward Propagation Process, image adapted from [31]

where ER_0 is the value of the expected reduction computed on the whole reference trajectory during the Backward Induction.

Once the new reference control policy and trajectory are selected, another Backward Induction can start and the whole procedure is repeated. The convergence is reached when the absolute value of the expected reduction $|ER_0|$ is less then a certain tolerance imposed by the user. Indeed the expected reduction is a quantity that gives an indication about the gradients of the cost-function with respect to the control. Therefore having a small ER_0 means that there is a small minimization of cost-function between two consecutive iterations and the solution is close to the local minimum searched by the method.

3.3.3 Limitations

The algorithm presented in this section is the basic local version of DDP, as it was originally formulated by Mayne [15]. It is a very powerful tool in solving optimal control problems due to the motivations already explained and as also shown by Liao and Shoemaker in [29] [30]. However, as it stands, it also presents some limitations:

- The problem addressed shall be **unconstrained**: neither stage-constraints nor constraints on final state of the trajectory are taken into account in the formulation. This is not the typical case in problems of Orbital Mechanics. Indeed in general the final state of the trajectory is prescribed by the target orbit and there may be some stage-constraints, such as the maximum deliverable thrust achievable by the engine of the spacecraft.

- The problem shall be featured by **positive-definite Hessian**: along the whole tra-

jectory, the Hessian Q_{uu} shall be always positive-definite. This is not always true for highly non-linear problems, such as the ones faced in this work. In order to grant a minimization of the cost-function J some adjustments shall be undertaken whenever this eventuality arises.

3.4 Hybrid Differential Dynamic Programming

Hybrid Differential Dynamic Programming is an improvement of the basic version of DDP that combine robust mathematical techniques with a DDP formulation in order to overcome the limitations aforementioned.
Being this algorithm the most important source of information for the realization of the work presented, in this section a detailed overview on HDDP is provided.
The chapter is organised as follows: first the major innovations introduced by HDDP are reported, then a global description of the algorithm is provided and lastly its limitations are illustrated.

3.4.1 Improvements introduced by HDDP

In this section, all the major mathematical improvements adopted in the HDDP algorithm [27] that differentiate it from the basic version introduced by Mayne [15] are presented. Initially, the innovations brought about by the use of the trust region method will then be reported, which almost radically revises the way in which the algorithm is implemented; next, we will discuss how this algorithm handles the different types of constraints that may be present; and finally, we will discuss the new state transition matrix approach, which allows for a more flexible and suitable calculation of the partial derivatives required during the backward induction process.

Trust-Region

One of the major improvements of the HDDP algorithm is the adoption of the Trust-region method [14]. Its introduction upsets the classical formulation of the DDP method according to which backward sweep and forward propagation are completely separate. In fact, it consists of a robust mathematical technique that fulfils a dual purpose in the algorithm. The first consists of replacing the classical line search method in forward propagation with a technique that is recognised in the literature as being more numerically stable. The second consists in introducing an efficient method capable of guaranteeing the positive definite condition of the Hessian, solving the shift issue and providing great robustness.

The method is exploited to find the coefficients of the feedback control variation as previously done in 3.3.1. However, differently from what previously seen, Trust region adds directly a constraint on the maximum magnitude that the constant term A_k can have (the so-called trust region radius). In this way the control variation at each step is restricted

in a certain region and it is prevented from stepping "too far". Reasonably, if the trust region radius is sufficiently small, the quadratic approximation reflects in a proper way the behaviour of the objective function. This allows to directly impose in the backward sweep a control law that respects at priori the approximation imposed by second order expansion, avoiding multiple integration in the forward propagation. Otherwise, if the solution generated is not in the neighborhood of the reference one, the backward sweep process must be repeated by decreasing the trust region radius.

Operatively, the method consists in solving the following subproblem at each time step k, named as TRQP($J_{u,k}, J_{uu,k}, \Delta$), referring to the inputs required to define it:

$$\min_{\delta u_k}(J_{u,k}\delta u_k + \frac{1}{2}\delta u_k^T J_{uu,k}\delta u_k) \qquad \text{subjected to} \quad ||D\delta u_k|| \leq \Delta \tag{3.20}$$

where Δ is the current trust region radius and D is a positive definite scaling matrix. The latter must be defined on the basis of the problem addressed, and is of paramount importance when the problem is poorly scaled (the same variation in different variables affects the objective function in a very different way).

The solution of this subproblem is obtained by following algorithms presented in [14], but an overview on the possible solving methods are also present in [36]. Of the various possibilities, given the small size m of the control vector, in this work the designated method of resolution of TRQP is the one based on eigenvalue decomposition of the Hessian, and it is reported in the appendix A.
To be concise, what is important to underline is that each solving method, included the one mentioned above, consists in an iterative procedure that produces an adequate shift of $J_{uu,k}$, in order to obtain a solution that respect the magnitude constraint. Once defined the solution δu_k^* and the shifted hessian $\widetilde{J}_{uu,k}$, at the end the solution of the subproblem consists in:

$$\delta u_k^* = A_k = -\widetilde{J}_{uu,k}^{-1} J_{u,k} \tag{3.21}$$

The shifted $\widetilde{J}_{uu,k}$ is then exploited to define all the other coefficients that appears in the feedback control law.
Differently on what presented in 3.3.1, a problem can also contain constraints and static parameter w. The latter ones are time-independent parameters that affect the solution, and often there is the intention to minimise the cost function with respect to those as well. The presence of constraints instead, generally leads to an augmented objective function that depends directly on them by means of some multipliers λ, as will be better explained in the next sections. However, for the general case the objective function J is typically

expanded also with respect to these parameters, becoming:

$$\begin{aligned}\delta J_k \approx\ & ER_{k+1} + J_{x,k}^T\delta x_k + J_{u,k}^T\delta u_k + J_{w,k}^T\delta w + J_{\lambda,k}^T\delta\lambda + \frac{1}{2}\delta x_k^T J_{xx,k}\delta x_k \\ & + \frac{1}{2}\delta u_k^T \widetilde{J}_{uu,k}\delta u_k + \frac{1}{2}\delta w^T J_{ww,k}\delta w + \frac{1}{2}\delta\lambda^T J_{\lambda\lambda,k}\delta\lambda + \delta x_k^T J_{xu,k}\delta u_k \\ & + \delta x_k^T J_{xw,k}\delta w + \delta u_k^T J_{uw,k}\delta w + \delta x_k^T J_{x\lambda,k}\delta\lambda + \delta u_k^T J_{u\lambda,k}\delta\lambda + \delta w^T J_{w\lambda,k}\delta\lambda\end{aligned} \tag{3.22}$$

Then, in [27], in analogy with the previous definition of the feedback control variation, the feedback control law assume the form:

$$\delta u_k = A_k + B_k\delta x_k + C_k\delta w + D_k\delta\lambda \tag{3.23}$$

The additional coefficients are determined once the shifted hessian $\widetilde{J}_{uu,k}$ is retrieved by the trust-region subproblem solution. Hence, following the same philosophy of the unconstrained case, imposing that $\frac{d}{d\delta u_k}\delta J_k = 0$:

$$\begin{cases} A_k = -\widetilde{J}_{uu,k}^{-1} J_{u,k} \\ B_k = -\widetilde{J}_{uu,k}^{-1} J_{ux,k} \\ C_k = -\widetilde{J}_{uu,k}^{-1} J_{uw,k} \\ D_k = -\widetilde{J}_{uu,k}^{-1} J_{u\lambda,k} \end{cases} \tag{3.24}$$

Furthermore, having now new derivatives in the 3.22 with respect to 3.10, also additional equation shall be respected for the update rule. Indeed, at the equations presented in 3.17, also the following list shall be taken into account:

$$\begin{aligned} J_{xw,k}^* &= J_{xw,k} + B_k^T J_{uu,k} C_k + B_k^T J_{uw,k} + J_{ux,k}^T C_k \\ J_{x\lambda,k}^* &= J_{x\lambda,k} + B_k^T J_{uu,k} D_k + B_k^T J_{u\lambda,k} + J_{ux,k}^T D_k \\ J_{w,k}^* &= J_{w,k} + A_k^T J_{uu,k} C_k + A_k^T J_{uw,k} + J_{u,k}^T C_k \\ J_{ww,k}^* &= J_{ww,k} + C_k^T J_{uu,k} C_k + C_k^T J_{uw,k} + J_{uw,k}^T C_k \\ J_{w\lambda,k}^* &= J_{w\lambda,k} + C_k^T J_{uu,k} D_k + C_k^T J_{u\lambda,k} + J_{uw,k}^T D_k \\ J_{\lambda,k}^* &= J_{\lambda,k} + A_k^T J_{uu,k} D_k + A_k^T J_{u\lambda,k} + J_{u,k}^T D_k \\ J_{\lambda\lambda,k}^* &= J_{\lambda\lambda,k} + D_k^T J_{uu,k} D_k + D_k^T J_{u\lambda,k} + J_{u\lambda,k}^T D_k \end{aligned} \tag{3.25}$$

At this point, when all the coefficients are determined (if there are static parameter or constraints also their variation shall be determined), the trajectory is propagated one single time, and then a check on the neighborhood is performed in order to accept or reject the iterate produced. In HDDP, the following parameter ρ is defined:

$$\rho = \frac{J_{new} - J}{ER_0} \tag{3.26}$$

Whenever the expected reduction well reflects the effective variation of the objective function (so the second order approximation is good), $\rho \approx 1$. This is why the following criterion is exploited in HDDP to accept the iterate:

$$\begin{cases} 1 - \epsilon_1 \leq \rho \leq 1 + \epsilon_1 & \text{iterate accepted} \\ \text{otherwise} & \text{iterate rejected} \end{cases} \tag{3.27}$$

where ϵ is a small parameter > 0, defined by the user (typically around 0.01).
Should the iteration be accepted or not, the trust region radius is modified accordingly. If the iteration is successfully accepted, the radius is increased in order to try to speed up the convergence rate. Conversely, when the iteration is rejected, this parameter is reduced in order to obtain a trajectory closer to the reference one. The updating rule proposed is the following:

$$\Delta_{k+1} = \begin{cases} \min((1+\kappa)\Delta_k, \Delta_{max}) & \text{if } \rho \in [1-\epsilon_1, 1+\epsilon_1] \\ \max((1-\kappa)\Delta_k, \Delta_{min}) & \text{otherwise} \end{cases} \tag{3.28}$$

where $0 < \kappa < 1$ is a constant selected at priori.

Constraints Handling

With respect to the basic version of DDP, HDDP can manage effectively also the presence of constraints. In this context, some improvements had already been made in previous works, like in Yakowitz [33] or Dyer and McReynolds [16]. The work of Lantoine and Russel takes some tried and tested techniques and uses them for the effective treatment of constrained problems.

First of all, a differentiation is made between two families, i.e. stage constraints and final state constraints, which are handled differently by the algorithm.
The first class are guaranteed to be satisfied to the first order with each iteration generated, while in the second case, an iterative method is generated that progressively brings the solution to comply with the constraint.

Range-Space Active Set Methods (stage constraints)

In order to handle stage constraints a Range-Space active set method is employed in both version of HDDP. It is based on a procedure that identifies if the constraint is active at the current step k. When this occurs, the latter is linearized and a constrained quadratic programming technique based on Fletcher's work [36] is then applied. This ensure the compliance with the constraints at the first order approximation.
The idea consists in computing initially the control variation δu_k^* in the same way of the unconstrained case, reported in section 3.4.1. Then the control $u_k = \bar{u}_k + \delta u_k^*$ is computed, and is checked if $g(\bar{x}_k, u_k, \bar{w}) \leq 0$. The active constraints identified are then collected in a matrix $\widetilde{g}$ and linearized.
Afterwards the control law is found by solving the minimization of the quadratic approximation of J_k subjected to the equality constraints:

$$\widetilde{g}_{u,k}^T \delta u_k + \widetilde{g}_{x,k}^T \delta x_k + \widetilde{g}_{w,k}^T \delta w_k + \widetilde{g}_c^T = 0$$

The method proposed by Fletcher consists in the fulfilment of the Karush-Kuhn-Tucker [7] conditions by defining the following Lagrangian function:

$$\begin{aligned}\mathcal{L}_k =& ER_{k+1} + J_{x,k}^T \delta x_k + J_{u,k}^T \delta u_k + J_{w,k}^T \delta w + J_{\lambda,k}^T \delta \lambda + \frac{1}{2} \delta x_k^T J_{xx,k} \delta x_k \\ &+ \frac{1}{2} \delta u_k^T \widetilde{J}_{uu,k} \delta u_k + \frac{1}{2} \delta w^T J_{ww,k} \delta w + \frac{1}{2} \delta \lambda^T J_{\lambda\lambda,k} \delta \lambda + \delta x_k^T J_{xu,k} \delta u_k \\ &+ \delta x_k^T J_{xw,k} \delta w + \delta u_k^T J_{uw,k} \delta w + \delta x_k^T J_{x\lambda,k} \delta \lambda + \delta u_k^T J_{u\lambda,k} \delta \lambda + \delta w^T J_{w\lambda,k} \delta \lambda \\ &+ \nu_k^T (\widetilde{g}_{u,k}^T \delta u_k + \widetilde{g}_{x,k}^T \delta x_k + \widetilde{g}_{w,k}^T \delta w_k + \widetilde{g}_c^T)\end{aligned} \tag{3.29}$$

Imposing that the derivatives of $\mathcal{L}_k$ with respect to the control policy variation and the lagrangian multipliers ν are equal to zero, the system 3.30 is obtained. Once solved, the coefficients of the new control policy are finally retrieved, Eq. 3.32.

$$\begin{bmatrix} \widetilde{J}_{uu,k} & \widetilde{g}_{u,k} \\ \widetilde{g}_{u,k}^T & 0 \end{bmatrix} \begin{bmatrix} \delta u_k \\ \nu_k \end{bmatrix} = \begin{bmatrix} -J_{u,k} - J_{xu,k}^T \delta x_k - J_{uw,k} \delta w - J_{u\lambda,k} \delta \lambda \\ -\widetilde{g}_c - \widetilde{g}_{x,k}^T \delta x_k - \widetilde{g}_{w,k}^T \delta w_k \end{bmatrix} \tag{3.30}$$

$$\delta u_k = A_k + B_k \delta x_k + C_k \delta w + D_k \delta \lambda \tag{3.31}$$

$$\begin{cases} G = (\widetilde{g}_{u,k}^T \widetilde{J}_{uu,k}^{-1} \widetilde{g}_{u,k})^{-1} \widetilde{g}_{u,k}^T \widetilde{J}_{uu,k}^{-1} \\ K = \widetilde{J}_{uu,k}^{-1} (I_m - \widetilde{g}_{u,k} G) \\ A_k = -K J_{u,k} - G^T \widetilde{g}_c \\ B_k = -K^T J_{xu,k}^T - G^T \widetilde{g}_{x,k}^T \\ C_k = -K^T J_{uw,k} - G^T \widetilde{g}_{w,k}^T \\ D_k = -K^T J_{u\lambda,k} - G^T \widetilde{g}_{\lambda,k}^T \end{cases} \tag{3.32}$$

Augmented Lagrangian (final state constraints)

In order to handle final state constraints (or for a multi-phase problem the so-called phase constraints) an augmented Lagrangian approach is employed [19]. The idea behind this approach is to consider an augmented objective function, that include two additional terms.

The first one is a "penalty term", which consists in the product between a scalar $\sigma > 0$ and the squared norm of the final state constraint's violation. Considering the objective function augmented by this term and following the same procedure as for unconstrained problems, gives rise to what in literature are known as penalty methods [19]. The addition of the penalty term allows to generate a solution that take into account also the constraints, limiting their violation. Once convergence is achieved, the greater the penalty parameter σ, the closer the solution will be to the imposed final state. However, this family of methods has a drawback. Indeed, if the penalty term is added to the objective function individually, in order to reach a solution that matches perfectly the prescribed final state, theoretically a parameter $\sigma = +\infty$ is required. This in practice is an unachievable condition, but nevertheless, in order to obtain a solution featured by a very small constraints violation, an high value of σ is needed. In this context the issue is that having a parameter that takes on such great values significantly alters the shape of the objective function. The consequence consists in the generation of numerical ill-conditioning that makes the convergence achievement particularly difficult for the algorithm.
Augmented Lagrangian methods overcome this problem by adding another term represented by the scalar product of a vector of multipliers λ and the vector of the constraint violation (recalling in this way a sort of Lagrangian function of the augmented objective function). This approach acts on modification not only of the penalty parameter σ but also of the vector of the multipliers λ. It has been demonstrated that this technique allows to reach a solution featured by a very small constraints violation also for limited values of σ and λ differently from penalty methods. In this way, ill-conditioning problems are prevented and the convergence rate is improved.
Operatively, in order to practically apply this approach in the HDDP algorithm, the final state cost function ϕ is substituted by the expression:

$$\widetilde{\phi} = \phi + \lambda^T \psi + \sigma ||\psi||^2 \tag{3.33}$$

In this way, exploiting the augmented cost function $\widetilde{\phi}$, the final state constraint is automatically included in the objective function expression.

At this point the algorithm procedure need some new passages with respect to the ones expressed in 3.3.1. In fact, an augmented Lagrangian method generally presupposes the modification of the multipliers vector during the solving process by following a precise law (either by exploiting the estimation at the first-order of the multipliers, or more refined approximations).
There are two version of HDDP, where the vector of multipliers is changed differently according to the approach adopted:

- In the first version of HDDP [28], the problem is faced by a "min max" approach. Here the detailed theoretical explication is omitted for conciseness, for a detailed description of this approach the reader can refer to the book of Bertsekas [6]. What is important to emphasise, however, is that in this approach, optimisation is performed by means of two separate cycles. In the first, a minimisation of the objective function is performed as in the unconstrained case, keeping the multipliers constant. Once the first minimisation is complete, the second cycle is executed just once, in which J is also expanded with respect to the multipliers. Their variation is computed imposing that $\frac{d}{d\delta\lambda}\delta J_0 = 0$, in a similar way to what have been done for the control, but, unlike the first loop, in this case the optimization consists in a maximisation with respect to the multipliers. Then, the $\delta\lambda$ computed take also part in the definition of the new control policy and the vector λ is changed accordingly for the next iteration.

- In the final version of HDDP presented in [27], the main idea behind the multipliers update is almost identical to what have been said for the previous case. The main difference consists in the fact that the optimization is performed in a single coupled cycle, where at each minimisation with respect to the control also a maximization with respect to the multipliers is "simultaneously" computed.

Operatively, in the cycle where there is the multipliers update, the idea consists in expanding J_k also with respect λ, obtaining the expansion 3.22. Then, retrieved all the coefficients with the procedure described in 3.4.1 I or in 3.4.1 III, an adequate variation of the multipliers shall be defined. In order to do that, in HDDP the hereunder rule is imposed, following the same philosophy of what previously done with the control:

$$\delta\lambda = A_\lambda + C_\lambda \delta w \tag{3.34}$$

with:

$$\begin{cases} A_\lambda = -\widetilde{J}_{\lambda\lambda}^{-1} J_\lambda \\ C_\lambda = -\widetilde{J}_{\lambda\lambda}^{-1} J_{\lambda w} \end{cases} \tag{3.35}$$

In [27], $\widetilde{J}_{\lambda\lambda}$ is defined solving the trust region subproblem TRQP$(-J_{\lambda,k}, -J_{\lambda\lambda,k}, \Delta)$. Differently from the one defined for the control, the solving process consists in a constrained maximization in $\delta\lambda$, no longer in a minimization.
As a final step of the backward process, the expected reduction of J is modified to take into account the increase in the objective function due to the maximisation on Lagrange multipliers

$$ER_0 = ER_0 + J_\lambda^T A_\lambda + \frac{1}{2} A_\lambda^T J_{\lambda\lambda} A_\lambda \tag{3.36}$$

Once the procedure to accept the iterate is then performed, it can sometimes happen that the norm of the final state constraints violation increases instead of decreasing. If it is the case, Lantoine and Russel proposed a rule for increase in a proper way the penalty

parameter σ, in order to push the next iterates towards feasibility. The updating rule proposed is the following:

$$\sigma_{k+1} = \max\left(\min\left(0.5\frac{h}{f^2}, k_\sigma \sigma_k\right), \sigma_k\right) \tag{3.37}$$

where k_σ is a constant greater than 1, and h, f are defined in the following way:

$$h = \phi(x_{N+1}, w) + \sum_{k=1}^{N} L(x_k, u_k, w) \tag{3.38}$$

$$f = \sqrt{||\psi(x_{N+1}, w)||^2} \tag{3.39}$$

At this point, due to the way in which these constraints are treated, the termination condition of algorithm is obtained when both two different conditions are matched:

- $||ER_0|| \leq \epsilon_{opt}$, which is the condition related to the gradients of the objective function, i.e. the closeness to the local minimum sought by the method.
- $||\psi(x_{N+1}, w)|| \leq \epsilon_{feas}$, which is the condition related to the closeness to the target final state.

ϵ_{opt} and ϵ_{feas} are two tolerances defined by the user, depending on the level of accuracy required and on the problem addressed.

State Transition Matrix approach

An other important aspect introduced by the HDDP is the State Transition Matrix (STM) approach [27]. This is a new way to compute the derivatives needed by the algorithm during the backward sweep, indeed they can be exploited in order to map the perturbations between two consecutive stages. The STMs substitute the derivatives of the transition function F that appear in the formulation. For problems that require integrated dynamics, i.e. where we have the dynamics f expressed as a set of differential equations, it is sometimes difficult, or downright impossible, to trace back to a transition function F analytically. The latter is in fact derived by approximating the integral of the dynamics f numerically (recalling the expression 3.3). This does not constitute a major problem in the calculation of the trajectory, however, in the computation of the derivatives of F it introduces a difficulty. In fact, changing the numerical integration method by which F is obtained also means changing its expression to be derived, which is exploited by the optimisation. Every time the method of integration is changed, in some way the derivatives of F are modified, thus also affecting the optimisation. This is why integration and optimisation, up to this point,

where considered somewhat linked in the DDP method and their coupling must be handled appropriately. The use of the STM fixes this problem, decoupling the integration by the optimization.

In the HDDP, an augmented system is exploited to compact the notation, in which the augmented state vector $X_k^T = [x_k^T \ u_k^T \ w^T]$ is considered. Accordingly, also an augmented transition function is taken into account $\widetilde{F}_k^T = [F_k^T \ 0_m \ 0_p]$ (since $\dot{u}_k = 0$ and $\dot{w} = 0$). By definition, the augmented system leads to:

$$X_{k+1} = \widetilde{F}_k(X_k) \tag{3.40}$$

$$\delta X_{k+1} \approx \ \widetilde{F}_{X,k}^T \delta X_k + \frac{1}{2} \delta X_k^T \bullet \widetilde{F}_{XX,k}^T \delta X_k = \Phi_k^1 \delta X_k + \delta X_k^T \bullet \Phi_k^2 \delta X_k \tag{3.41}$$

where Φ_k^1 represents the state transition matrix of order 1 of the augmented system, whereas Φ_k^2 is the state transition matrix of order 2.
These two tensors can be provided directly by the user, can be derived directly by the F (whenever there is an analytical form of the transition function), or can be retrieved by a numerical integration that requires just the knowledge of the dynamics expressed as f. Indeed, in this latter case, the state transition matrices can be computed by the integration of the following differential system:

$$\begin{cases} \dot{\Phi}_k^1 = f_X \Phi_k^1 \\ \dot{\Phi}_k^2 = f_X \bullet \Phi_k^2 + \Phi_k^{1T} \bullet f_{XX} \bullet \Phi_k^1 \end{cases} \tag{3.42}$$

subjected to the initial condition $\Phi_k^1(t_k) = I_{n+m+p}$ and $\Phi_k^2(t_k) = 0_{n+m+p}$.
The last casuistry is typically exploited in the case where there is an integrated dynamic in which an F function cannot be easily traced analytically. With this new approach, it should be emphasised that the derivatives of F are obtained by means of numerical integration, which requires knowledge only of the dynamics f, without the need of derivating a complex expression of a numerical integration scheme by which the F will be expressed. In this way the integration does not affect directly any more the optimization, and the user is facilitated to make the inputs of the problem faced conform to the optimisation algorithm.

Once the STMs are computed, substituting 3.41 into the expansion of J_{k+1}^*, the equation to compute the derivatives of J at the previous stage can be derived, in analogy with the process expressed in 3.3.1. The rule is here reported, considering the new STM approach introduced and the presence of static parameters w and final state constraints ψ.

$$\begin{bmatrix} J_{x,k} \\ J_{u,k} \\ J_{w,k} \end{bmatrix}^T = \begin{bmatrix} L_{x,k} \\ L_{u,k} \\ L_{w,k} \end{bmatrix}^T + \begin{bmatrix} J_{x,k+1}^* \\ 0_m \\ J_{w,k+1}^* \end{bmatrix}^T \Phi_k^1 \tag{3.43}$$

$$\begin{bmatrix} J_{xx,k} & J_{xu,k} & J_{xw,k} \\ J_{ux,k} & J_{uu,k} & J_{uw,k} \\ J_{wx,k} & J_{wu,k} & J_{ww,k} \end{bmatrix} = \begin{bmatrix} L_{xx,k} & L_{xu,k} & L_{xw,k} \\ L_{ux,k} & L_{uu,k} & L_{uw,k} \\ L_{wx,k} & L_{wu,k} & L_{ww,k} \end{bmatrix} + \Phi_k^{1T} \begin{bmatrix} J^*_{xx,k+1} & 0_{n\times m} & J^*_{xw,k+1} \\ 0_{m\times n} & 0_{m\times m} & 0_{m\times p} \\ J^{*T}_{xw,k+1} & 0_{p\times m} & J^*_{ww,k+1} \end{bmatrix} \Phi_k^1 + \begin{bmatrix} J^*_{x,k+1} \\ 0_m \\ J^*_{w,k+1} \end{bmatrix}^T \bullet \Phi_k^2 \tag{3.44}$$

$$J_{\lambda,k} = J^*_{\lambda,k+1} \qquad J_{\lambda\lambda,k} = J^*_{\lambda\lambda,k+1} \tag{3.45}$$

$$\begin{bmatrix} J^T_{x\lambda,k} & J^T_{u\lambda,k} & J^T_{w\lambda,k} \end{bmatrix} = J^T_{X\lambda,k} = J^{*T}_{X\lambda,k+1} \frac{\partial X_{k+1}}{\partial X_k} = \begin{bmatrix} J^{*T}_{x\lambda,k+1} & 0_m & J^{*T}_{w\lambda,k+1} \end{bmatrix} \Phi_k^1 \tag{3.46}$$

3.4.2 Limitations

Up to this point, the various improvements made by HDDP have been illustrated to demonstrate its potential. On the other hand, this algorithm is also subject to certain limitations that circumscribe its use and on which further improvements shall be investigated. The two major ones are related to the employment of the mathematical techniques that at the same time represent the major innovations introduced by HDDP: the trust region method and the STM approach.

STM Computations

The introduction of the STM approach provide several advantages. However, its calculation requires a lot of computational effort. In fact, the integration of the first and second order STMs requires $(n + m + p)^2$ and $(n + m + p)^3$ equation to be integrated. This implies computational cost that increase rapidly as the number of the variables of the problem increase. In this regard, many studies carried out using HDDP have shown that the part of the algorithm that takes the longest computational time is precisely related to the calculation of STMs. In comparison, the traditional Riccati-like formulation (implemented in the software Mystic [34]) requires only $(n+m+p)$ and $(n+m+p)^2$ equation to be integrated. This is why considerable attention is now being paid to studies that aim to decrease the computational time dedicated to STMs, like for example numerical approximations of STMs [31] or the use of analytical STMs [20] that approximate the actual dynamics whenever its possible.

Tuning of the Algorithm

An other open points is related to the tuning of HDDP. Indeed, in the algorithm there are a lot of parameters to be tuned at priori in order to grant/speed up convergence. One of the most important parameters to be tuned in this sense, is the scaling matrix D of the trust-region method. The latter is of paramount importance for a convergence of the algorithm, especially when the problem faced presents some final state constraints. In this case indeed, also if the problem is well-scaled (so in general a good nondimensionalization is performed and the same variation on different variables produces similar variation on the objective function), typically the variation brought about by the lagrangian multipliers λ is very different by the one brought about by the state variables. This is why, in any case, D_u and D_λ are different. Moreover, there is no an automatic rule to rely on in order to define a scaling matrix that ensure the convergence. This implies that some tests are required to define the D matrix in a proper manner, in order to grant convergence.
However, this tuning operation can sometimes be quite complicated and difficult to resolve effectively in practice.

3.4.3 HDDP Algorithm

Step 0 (Initialization): Assume an initial guess for dynamic controls u_k and Lagrange multiplier λ (tipically $\lambda = 0$). Define all the initial values of optimization parameters, i.e. initial trust region radius Δ_0, initial penalty parameter σ_0, convergence thresholds $\epsilon_{opt}, \epsilon_{feas}$, and constants parameters like D matrix, $\kappa, k_\sigma, \epsilon_1$. Compute trajectory, initial objective and constraints violation values.

Step 1 (Computation of first/second order STMs): Evaluation of ϕ_k^1 and ϕ_k^2 for all the stages k. If this process is performed by the integration of 3.42, this is the most computational expensive step of the algorithm.

Step 2 (Backward Sweep): Initialize the optimal cost-to-go function derivatives and the expected reduction ER at the last stage. Perform recursive mapping of control, state and multiplier cost derivatives through 3.43, 3.44, 3.45 and 3.46. Solve the successive trust region subproblems of 3.22. Deduce the control law coefficients A_k, B_k, C_k and D_k for δu_k from 3.24 (unconstrained case) or 3.32 (constrained case); the control law coefficients A_λ and C_λ for $\delta\lambda$ from 3.35. Compute the total expected reduction ER_0 from repeated application of 3.17 (first equation) and 3.36.

Step 3 (Convergence Test): if $ER_0 < \epsilon_{opt}, f < \epsilon_{feas}$, all reduced Hessian J_{uu} with respect to u are positive definite and Hessian $J_{\lambda\lambda}$ with respect to λ is negative definite, then STOP (CONVERGENCE ACHIEVED).

Step 4 (Forward Sweep): Compute a new trial iterate with the control laws obtained from Step 2. Evaluate J_{new}, h_{new} and f_{new}.

Step 5 (Trust region Update and Acceptance of an Iteration): Compute the cost ratio ρ from 3.26. Update trust region radius Δ following the rule 3.28. If $\rho \in [1-\epsilon_1, 1+\epsilon_1]$ go to Step 6, otherwise go to Step 2.

Step 6 (Penalty Update): if $f_{new} > f$, update the penalty parameter using 3.37.

Step 7 (Nominal Solution Update): Replace the values of J, h, f, u_k, w, λ and x_k by their new values. Go to Step 1.

4 | HDDP in Orbital Elements

This chapter reports on the major contribution made by this thesis work. It presents the coupling of the optimisation algorithm discussed in the previous chapter with the use of orbital elements as a set of variables.
The structure of the optimiser remains the same, however, attention must be paid to certain aspects due to the intrinsic nature of the new set of variables, as also mentioned in [32] and [12]. In this chapter, we will begin by introducing the reader to a brief overview of what orbital elements are and what they represent, followed by a discussion of the advantages and the necessary precautions of using such variables in the context of HDDP, to finally move on to present a method for change the independent variable in the formulation, the Sundman transformation, which has considerable numerical benefits for certain types of problems addressed.

4.0.1 Reference frames

Before to start the overview it is important to define the 3 reference frame in which the states typically work. The Cartesian coordinates are in general defined in a fixed inertial x, y, z framework that can be for example Earth-Centered Inertial (ECI) or International Celestial Reference Frame (ICRF). However, sometimes can be convenient to define the states in other framework not fixed at priori. The two most common are Perifocal (EPH) one and the TNH [21]. EPH has x-axis that points towards the pericentre location, z-axis that is in the same direction of the angular momentum of the orbit and the y-axis that complete the right-handed tern. TNH, instead, is the frame that has y-axis directed as the conjunction between the point considered and the focus of the orbit, the z-axis directed as the angular momentum and x-axis that complete the right-handed tern.

4.1 Orbital elements overview

There are two main sets of orbital elements that are usually used to deal with orbital mechanics problems, both of which allow to reformulate the problem no longer in terms of the classical Cartesian coordinates, but as a function of certain parameters of the orbit on which an hypothetical spacecraft lies at certain instant. Sometimes, such a representation takes on major importance in this context, since it allows one to evaluate how an orbital transfer takes place in terms of specific feature of an orbit, like the dimension and the shape, also being able to interpret which part of the transfer is most energy-consuming, which is difficult to interpret when the user can observe trends in Cartesian coordinates.
The first set consists of the so-called Keplerian elements, which have a close correlation with typical physical and geometric quantities of the orbit. In the literature they are typically reported with as a, e, i, Ω, w, θ of which:

- $\boldsymbol{a}$ is called *semimajor axis*. It represents the dimension of the orbit. The last quantity is also closely related to the energy level, and therefore this parameter gives important information in this regard as well.
- $\boldsymbol{e}$ is called *eccentricity*. It represents the shape of the orbit. Indeed, in the two-body environment, the orbit is represented by a conic section, i.e. an ellipses, a parabola or an hyperbola. This parameter indicates to which family the orbit belongs and its ellipticity.
- $\boldsymbol{i}$ is called *inclination*. It represent the angle between the normal to the orbital plane and the direction of the z-axis in the Cartesian reference system.
- $\boldsymbol{\Omega}$ is called *right-ascension of the ascending node*, which is the angle between the x-axis in the Cartesian reference frame and the direction defined by the intersection of the orbital plane on the xy-plane of the same latter framework (which can be called node direction).
- $\boldsymbol{w}$ is called *anomaly of pericentre*. It is the angle between the direction defined by the intersection of the orbital plane on the xy-plane in the Cartesian reference frame and the one defined by the orbital pericentre location.
- $\boldsymbol{\theta}$ is called *true anomaly*. It represents the angle between the pericentre direction and the point in which the spacecraft lies on the orbit.

This set of variables can be retrieved directly the the Cartesian expression of the state by applying some non-linear transformations that are well known in orbital mechanics. To see a complete view of these the reader can refer to [21].

The second widely used set refers to the so-called Modified-equinoctial elements. They are reported in literature as p, f, g, h, k, L, which:

- $\boldsymbol{p}$ is called *semilatus rectum* and represents the distance by the attractor when $\theta = 90°$. It still has similar meaning of the semi-major axis a in terms of energy and dimension of the orbit, but it has the property that for any value of eccentricity e it never diverges neither change sign. The semi-major axis instead is $= \infty$ when the orbit become a parabola and is less than 0 when the latter becomes an hyperbola.
- $\boldsymbol{f, g}$ represents the x and y components of the eccentricity vector in the EPH frame.
- $\boldsymbol{h, k}$ represents the x and y components of the node vector in the orbital frame.
- $\boldsymbol{L}$ is called *true longitude*

The last set can be retrieved directly from the knowledge of the classical Keplerian elements and viceversa, by applying the following transformations:

$$\begin{aligned} p &= a(1-e^2) \\ f &= e\cos(\Omega+\omega) \\ g &= e\sin(\Omega+\omega) \\ h &= \tan\left(\tfrac{i}{2}\right)\cos(\Omega) \\ k &= \tan\left(\tfrac{i}{2}\right)\sin(\Omega) \\ L &= \Omega+\omega+\theta \end{aligned} \tag{4.1}$$

$$\begin{aligned} a &= \frac{p}{1-e^2} \\ e &= \sqrt{f^2+g^2} \\ i &= 2\tan^{-1}(\sqrt{h^2+k^2}) \\ \Omega &= \tan^{-1}(k,h) \\ \omega &= \tan^{-1}(g,f) - \Omega \\ \theta &= L - (\Omega+\omega) \end{aligned} \tag{4.2}$$

where equation 4.1 represents the direct transformation, whereas equation 4.2 represents the inverse one. tan^{-1} indicates a four quadrant inverse tangent calculation.

4.2 Advantages and precautions

This section discusses in detail all the advantages and precautions required for a successful optimisation method when adopting orbital elements as a set of state variables.

4.2.1 Precautions

The two main points to pay attention to are the scaling process, which as we will see is crucial here, and some pros and cons associated with the different choice of set adopted for the problem addressed.

Scaling Process

As seen in the section 4.1, orbital elements presents as state variables of different nature. Indeed, a or p consists in physical distances, i, Ω, w, θ, L are geometric angles, e is a dimensionless parameter and f, g, h, k are quantities that are difficult to label.
In particular, it should be emphasised that for this reason, some quantities have dimensions that are significantly different from others. This is the case for the semi-major axis and the semilatus rectum, which have typical values that exceed tens of thousands of kilometres, and other elements, which, being geometric angles or projections of almost unitary vectors, are bounded to values just over the unity. This phenomenon causes an inherent ill-conditioning of the matrices involved in the treatment and consequently a scaling process becomes of paramount importance for the success of the optimisation in this case. This aspect is a good practice also when Cartesian coordinates are exploited, but for that case sometimes a convergence can be achieved also without the scaling process. For orbital elements instead, it becomes fundamental to avoid intrinsic ill-conditioning due to the reasons aforementioned.

In this work to avoid numerical problems due to this aspect the following characteristic quantities are adopted, which are similar to the ones proposed by [12], i.e.:

- $L_{ref} = a_{target}/1.5$, reference length equal to target semi-major axis divided by a factor equal to 1.5;
- $m_{ref} = m_0$, reference mass equal to initial total mass of the spacecraft;
- $t_{ref} = \sqrt{\frac{L_{ref}^3}{\mu}}$, reference time chosen to have some coefficients in the dynamic equation equal to 1;
- $u_{ref} = \frac{\mu\, m_{ref}}{L_{ref}^2}$, reference thrust chosen to have some coefficients in the dynamic equation equal to 1;

The factor equal to 1.5 that appear in the espression of L_{ref} is introduced to ensure that the solution, when close to convergence, will have a dimensionless semi-major axis equal to the factor. It was chosen greater than 1 in order to give slightly more weight to the violation of the constraint on the semi-axis than the other parameters. Indeed, it is noted during optimization that a variation of a leads also a variation of ω and θ (because the optimization acts on u_t). They are the variables in the problem that exhibit the most oscillatory

behaviour, which means that the greatest variation typically occurs on them. It is therefore sometimes the case that the method focuses more on reducing the final constraint violation associated with these angles, having difficulty in optimising the semi-axis efficiently at the same time. At the end of this procedure the parameters that are rescaled are: the semi-major axis a, the mass m, the components of the thrust T_t, T_n, T_h, the specific impulse I_{sp}[1], the constant acceleration g_0 and the time of flight TOF selected for the transfer.

The trajectory is generated integrating dimensionless Gauss planetary equations reported in 4.3, which directly involve orbital elements as a variables. To this set of differential equations, the mass equation is added, obtaining the complete dynamics that describe the problem faced.

$$
\begin{cases}
\dot{\tilde{a}} = 2\sqrt{\dfrac{\tilde{a}^3}{1-e^2}(1+2e\cos\theta+e^2)}\left(\dfrac{\tilde{u}_t}{\tilde{m}}\right) \\
\dot{e} = \sqrt{\dfrac{\tilde{a}(1-e^2)}{1+2e\cos\theta+e^2}}\left[2(e+\cos\theta)\left(\dfrac{\tilde{u}_t}{\tilde{m}}\right) - \dfrac{(1-e^2)\sin\theta}{1+e\cos\theta}\left(\dfrac{\tilde{u}_n}{\tilde{m}}\right)\right] \\
\dot{i} = \sqrt{\tilde{a}(1-e^2)}\dfrac{\cos(\omega+\theta)}{1+e\cos\theta}\left(\dfrac{\tilde{u}_h}{\tilde{m}}\right) \\
\dot{\Omega} = \dfrac{1}{\sin i}\sqrt{\tilde{a}(1-e^2)}\dfrac{\sin(\omega+\theta)}{1+e\cos\theta}\left(\dfrac{\tilde{u}_h}{\tilde{m}}\right) \\
\dot{\omega} = \dfrac{1}{e}\sqrt{\dfrac{\tilde{a}(1-e^2)}{1+2e\cos\theta+e^2}}\left[2\sin\theta\left(\dfrac{\tilde{u}_t}{\tilde{m}}\right) + \dfrac{2e+\cos\theta+e^2\cos\theta}{1+e\cos\theta}\left(\dfrac{\tilde{u}_n}{\tilde{m}}\right)\right] \\
\qquad -\sqrt{\tilde{a}(1-e^2)}\dfrac{\cos i}{\sin i}\dfrac{\sin(\omega+\theta)}{1+e\cos\theta}\left(\dfrac{\tilde{u}_h}{\tilde{m}}\right) \\
\dot{\theta} = \dfrac{(1+e\cos\theta)^2}{\sqrt{\tilde{a}^3(1-e^2)^3}} - \dfrac{1}{e}\sqrt{\dfrac{\tilde{a}(1-e^2)}{1+2e\cos\theta+e^2}}\left[2\sin\theta\left(\dfrac{\tilde{u}_t}{\tilde{m}}\right) + \dfrac{2e+\cos\theta+e^2\cos\theta}{1+e\cos\theta}\left(\dfrac{\tilde{u}_n}{\tilde{m}}\right)\right] \\
\dot{\tilde{m}} = -\dfrac{\sqrt{\tilde{u}_t^2+\tilde{u}_n^2+\tilde{u}_h^2}}{\widetilde{I_{sp}g_0}}
\end{cases}
\tag{4.3}
$$

[1] $\widetilde{I_{sp}g_0} = I_{sp}g_0\sqrt{\frac{L_{ref}}{\mu}}$

As can be seen in the equations 4.3 is not present the gravitational parameter μ. This is precisely due to the scaling process.
At the same time, the dimensionless dynamics equation exploited when modified-equinoctial elements are adopted as state variables, are reported in 4.4, which are provided by [32]. For this work it has been decided to maintain the semi-major axis a rather than the semilatus rectum p. This is due to the fact that the transfers studied involve only elliptical orbits, thus avoiding the problems caused by the divergence/change of sign of this element. At the end of the scaling procedure (which is performed exploiting the same characteristic quantities adopted for the Keplerian elements evolution) the dynamics equation is represented by:

$$
\begin{cases}
\dot{\tilde{a}} = 2\sqrt{\tilde{a}^3}\sqrt{\dfrac{1+2f\cos L+2g\sin L+f^2+g^2}{1-f^2-g^2}}\left(\dfrac{\tilde{u}_t}{\tilde{m}}\right) \\
\dot{f} = \sqrt{\tilde{a}}\sqrt{\dfrac{1-f^2-g^2}{1+2f\cos L+2g\sin L+f^2+g^2}}\Bigg\{2(f+\cos L)\left(\dfrac{\tilde{u}_t}{\tilde{m}}\right) - \Bigg[2g+ \\
\quad \dfrac{1-f^2-g^2}{1+f\cos L+g\sin L}\sin L\Bigg]\left(\dfrac{\tilde{u}_n}{\tilde{m}}\right)\Bigg\} - g\sqrt{\tilde{a}(1-f^2-g^2)}\dfrac{h\sin L-k\cos L}{1+f\cos L+g\sin L}\left(\dfrac{\tilde{u}_h}{\tilde{m}}\right) \\
\dot{g} = \sqrt{\tilde{a}}\sqrt{\dfrac{1-f^2-g^2}{1+2f\cos L+2g\sin L+f^2+g^2}}\Bigg\{2(g+\sin L)\left(\dfrac{\tilde{u}_t}{\tilde{m}}\right) - \Bigg[2f+ \\
\quad \dfrac{1-f^2-g^2}{1+f\cos L+g\sin L}\cos L\Bigg]\left(\dfrac{\tilde{u}_n}{\tilde{m}}\right)\Bigg\} - f\sqrt{\tilde{a}(1-f^2-g^2)}\dfrac{h\sin L-k\cos L}{1+f\cos L+g\sin L}\left(\dfrac{\tilde{u}_h}{\tilde{m}}\right) \\
\dot{h} = \dfrac{1}{2}\sqrt{\tilde{a}(1-f^2-g^2)}\dfrac{1+h^2+k^2}{1+f\cos L+g\sin L}\cos L\left(\dfrac{\tilde{u}_h}{\tilde{m}}\right) \\
\dot{k} = \dfrac{1}{2}\sqrt{\tilde{a}(1-f^2-g^2)}\dfrac{1+h^2+k^2}{1+f\cos L+g\sin L}\sin L\left(\dfrac{\tilde{u}_h}{\tilde{m}}\right) \\
\dot{L} = \dfrac{(1+f\cos L+g\sin L)^2}{\sqrt{[\tilde{a}(1-f^2-g^2)]^3}} + \sqrt{\tilde{a}(1-f^2-g^2)}\dfrac{h\sin L-k\cos L}{1+f\cos L+g\sin L}\left(\dfrac{\tilde{u}_h}{\tilde{m}}\right) \\
\dot{\tilde{m}} = -\dfrac{\sqrt{\tilde{u}_t^2+\tilde{u}_n^2+\tilde{u}_h^2}}{\widetilde{I_{sp}g_0}}
\end{cases}
\tag{4.4}
$$

Choice of the set

The other precautions to pay attention is related to the choice of the set of orbital elements adopted.
All the classical Keplerian orbital elements has a strong physical meaning, representing dimension, shape and orientation in space of the current orbit. This implies that their evolution over time provides an easy-to-interpret view of how the transfer orbit is changes size, shape and how it rotates in space. However, this set, as can be seen by the dynamic Equation 4.3, is affected by several singularity conditions. In cases when $e = 0$, $e = 1$ and $i = 0$, which are often situations of interest for practical applications, the dynamic diverges, causing the failure of the method. Every time the path approaches these conditions, even without reaching them, the behaviour of the optimiser worsens, slowing down and sometimes completely preventing convergence.
Modified equinoctial elements on the other hand, has the advantage to do not present any singularity condition, but their evolution is not easy to interpret as they have no direct physical meaning (with the exception of p).

4.2.2 Advantages

The two main advantages that suggest the use of these variables' set are related to their evolution in time and the framework in which the dynamic works, that helps to define in a more easy way a better first guess.

Time evolution

It is well known in orbital mechanics that the orbital elements presents a behaviour which is quite different from the Cartesian coordinate representation. Considering for example the evolution of the two cases on a frozen keplerian orbit in the two-body environment. When Cartesian coordinates are exploits to describe this situation, all the 6 variable oscillates in a trend similar to a sinusoid. On the other hand, because the characteristic property of the orbit (i.e. dimension, shape and orientation of the orbital plane) does not change, a, e, i, Ω, w remain constant and the unique parameter that oscillates is the true anomaly. In this situation, therefore, the evolution of orbital elements exhibits a more regular and less oscillating behaviour than the Cartesian trend. In a general orbital transfer also the orbital elements changes, but their evolution in time is more smooth than the Cartesian ones in any case. With regard to the modified equinoctial elements, their behaviour is even more regular than the classical Keplerian elements, and they are considered the most efficient set in this respect.
This feature becomes very important in the optimisation context. In fact, from a numerical viewpoint, it facilitates the optimiser's task also with a limited amount of discretisation stages N. This results in optimisation taking place faster and with fewer iterations whenever the orbital elements are exploited as state variables.

First guess generation

Another important aspect related to the number of iterations required to achieve convergence is the definition of a good initial guess. Typically, when Cartesian coordinates are exploited as state variables, this is a issue to be solved. In fact results difficult to define a control law that generates the solution desired to the difficulty in identifying how a control law affect the characteristic properties of the orbit (energy, shape, orientation) when this set is employed. Morover, typically describing the problem in fixed refernce frame, like ECI, a good initial control policy require the variation in time of all the 3 components, which is difficult to be defined at priori.
Considering instead an orbital element's set in the TNH reference frame, the situation changes. In fact working directly on specif properties of an orbit, the dynamic equation directly provide what components of the thrust generates a variation in shape, dimension or orientation. In this way it is more easy to identify the control law required for the definition of a good initial guess. Moreover, working in the TNH reference frame, the a good initial control policy results more easy to be defined, because it can be retrieved maintaining costant the components along the whole path. For example, for a planar case, where initial and target orbit are circular and lies on xy plane in the ECI frame, a good initial guess is a trajectory that approximately match the final dimension of the target orbit. To do so, in TNH the issue is simply solved by defining a control policy which has just a constant tangent component, where the magnitude is the only thing to be tuned. Whenever instead the dynamic is represented in ECI, an oscillating control law of both x and y thrust component shall be defined, which can also change if the discretization number change. It is evident that in the latter case the issue is more cumbersome to be efficiently solved.
Therefore, this discussion underlines how is easier to define a good initial control law whenever the problem is formulated by means of orbital elements.

4.3 Challenges and Sundman Transformation

From the optimization point of view, when we are dealing with a planetary multi-revolution transfer, the problem becomes very sensitive to variations in control and oscillations in state variables. Moreover, due to its intrinsic nature, a large number of discretization stages N is needed to well describe these kind of transfers.
For these reasons, it is very difficult for the algorithm, if not almost impossible, to achieve convergence if classical Cartesian variables are used, due to their fluctuating behaviour. The use of orbital elements in these cases, given their more regular evolution over time, becomes not only a way to speed up convergence as in the previous study case, but also a decisive means for the success of the method.

Another important aspect to keep in mind is how the discretization is handled. In fact,

it can happen in this type of transfer that the number of nodes per revolution varies along the trajectory. This means that if the independent variable is discretised in a total number of nodes N in an equispaced manner, it may happen that for certain revolutions a much greater (or lesser) number of nodes is found than for others. Furthermore, even on a single revolution, depending typically on the shape of the orbit, there may be an accumulation of nodes in certain areas rather than others.
This is the typical case when time t is used as independent variable, as has been done so far in the previous case studies. It is well known that as the semi-major axis a increases, the period increases, thus leading to an accumulation of nodes as this element grows during the transfer and a poor discretization when this parameter is small and the dynamics is faster. Similarly, for eccentric orbits where the velocity along the path is not constant, on a single revolution, an accumulation of nodes in the apoapsis area in spite of the periapsis occurs. These inhomogeneities in the spacing between nodes into which the path is divided make the optimisation performed by the algorithm more inefficient and difficult, increasing the number of iterations required to reach convergence.

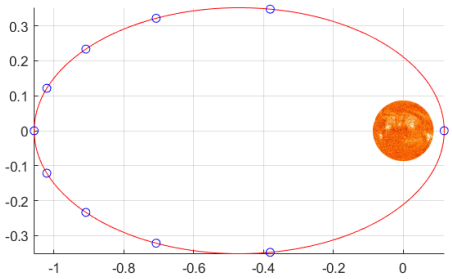
In order to solve these issues, it is convenient to rewrite the problem in terms of a new independent variable, leading to a different discretization from the one described. This procedure is carried out by introducing the Sundman's transform [23], which allows everything to be reformulated as a function of an angle, such as the eccentric anomaly E or the true anomaly θ.
The transformations that allow the dynamics to be rewritten as a function of the new variables are the ones reported in [32]:

$$\begin{aligned} dt &= \sqrt{\frac{a^3}{\mu}} \frac{1-f^2-g^2}{1+f\cos L+g\sin L} dE \\ dt &= \sqrt{\frac{a^3}{\mu}} \frac{(1-f^2-g^2)^{3/2}}{(1+f\cos L+g\sin L)^2} d\theta \end{aligned} \tag{4.5}$$

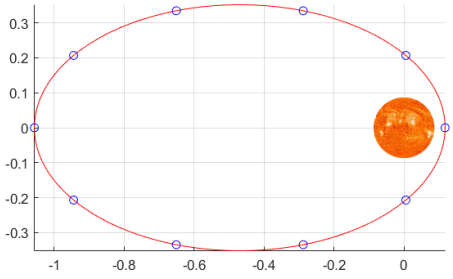
At this point the dynamics equation can be reformulated considering:

$$\frac{d\,[a,f,g,h,k,L]^T}{d\nu} = \frac{d\,[a,f,g,h,k,L]^T}{dt}\frac{dt}{d\nu} \tag{4.6}$$

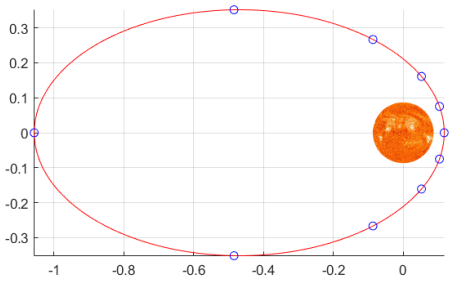
where ν represents the generic angle in which the dynamic is rewritten (which can be E either θ) and the derivatives $\frac{dt}{d\nu}$ is recovered by 4.5.
At this point, spacing is handled differently depending on the choice of the new independent variable.



(a) Time



(b) Eccentric Anomaly



(c) True Anomaly

Figure 4.1: Different discretizations of an orbit with eccentricity e=0.8, depending on the independent variable selected

First of all, it must be emphasised that in the case of both eccentric anomaly and true anomaly as a new independent variable, the dependence of the spacing as a function of the semi-major axis is missing. In fact, by expressing everything as a function of a geometric angle, the discretization no longer presents accumulations and dispersions according to the different value of the orbital period, as was the case with time.
The only dependence that remains is on the shape, i.e. the eccentricity of the orbit, which changes from case to case.
When time is selected, as previously anticipated, there is an accumulation in proximity of the apoapsis, and this represents the worst case from the optimizer point of view. In the case of the true anomaly, on the other hand, the situation is the opposite. Indeed, there is accumulation in the vicinity of the periapsis and dispersion in the area of the apoapsis.

Finally, as far as the eccentric anomaly is concerned, we find the greatest possible uniformity in the discretisation, where the latter also loses its dependence on eccentricity, obtaining perfectly equispaced nodes in all eventualities.
From this point of view, the use of the eccentric anomaly seems to be the best choice for the description of a multi-revolution problem. However, as later numerical tests will confirm, similar results are also obtained with the true anomaly, despite the fact that it presents a non-uniformity in spacing. This phenomenon can be attributed to the nature of the orbital dynamics, which is faster at the peripsis. In this area, therefore, it seems reasonable to assume a greater oscillation on the control as θ varies. The use of the true anomaly as an independent variable therefore generates a distribution of nodes that, although not uniform, is adapted to the speed at which the dynamics evolves along the trajectory. In conclusion, both choices constitute a good alternative to the use of time as an independent variable in order to extend the number of revolutions that can be treated in the optimisation, speed up the achievement of convergence and limit the number of discretization nodes into which the path will be divided.

5 | Results

5.1 Direct Transfer in Continuous Thrust

5.1.1 Description of the study case

As a first practical application, a transfer between two prescribed orbits in a planetary environment was selected as a case study. The trajectory is imposed as a direct transfer without revolutions and the dynamic selected is a Keplerian two-body model with the Earth as a main attractor. The case under investigation aims to test the capability, on a preliminary basis, of the optimization algorithm to deal effectively with problems formulated in terms of orbital elements, evaluating performances and any limitation that may arise.

The state vector of the problem is represented by the 6 classical orbital elements and the mass of the spacecraft, whereas the control inputs consists in the 3 components of the thrust, expressed in the TNH reference frame.

In order to test the reliability and the robustness of the algorithm, initial and target orbits are specially selected with significantly different orbital parameters. Indeed they are:

	a [km]	e [-]	i [deg]	Ω [deg]	ω [deg]	θ [deg]
$\mathbf{x}_0$	21378	0.4	5	0	0	60
$\mathbf{x}_{target}$	42378	0.1	72	72	72	145

Table 5.1: Initial and Target orbital elements of the direct transfer

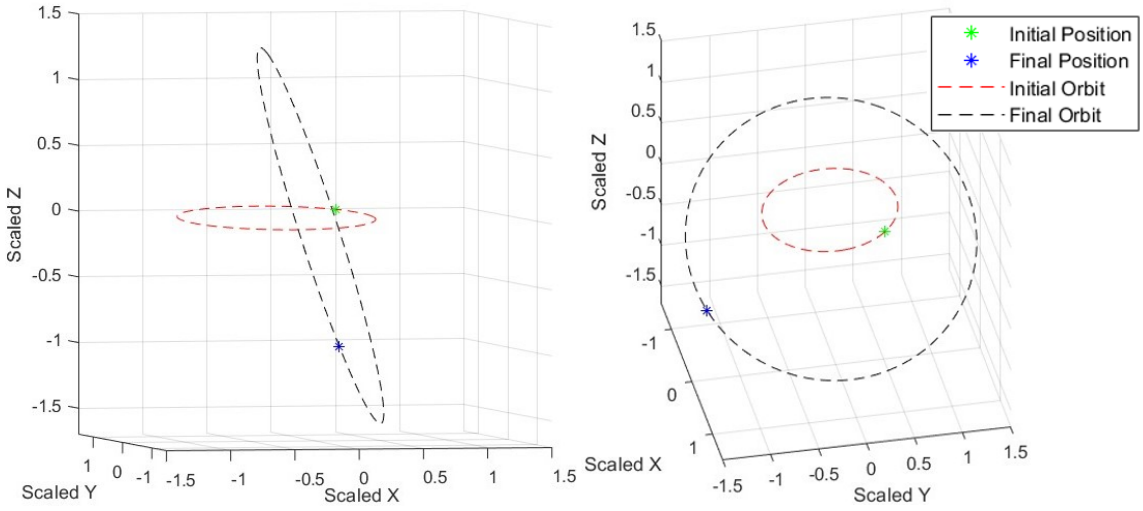


Figure 5.1: Different views of Initial and Target Position and Orbits in scaled 'x,y,z' frame

As can be seen from figure 5.1, between the two orbits there is not only a change in the semi-major axis and in eccentricity, but also a remarkable change in the orbital plane. In this way the performances are evaluated when it is required a significant modification in all the state variables in order to match the final target, stressing the optimization capabilities of the algorithm.

The objective function J to be minimized is selected as the energy integral and the final state constraints are set in a classical linear manner. Due to the nature of transfer, a number of stages $N = 50$ was considered sufficient for the problem at hand.

$$J = \sum_{k=1}^{N} ||\tilde{\mathbf{u}}_k||^2 \Delta\tilde{t}_k \qquad \text{with } \Delta\tilde{t}_k = \tilde{t}_{k+1} - \tilde{t}_k \tag{5.1}$$

$$\psi = \left[\tilde{a}_0 - \tilde{a}_{target},\ e_0 - e_{target},\ i_0 - i_{target},\ \Omega_0 - \Omega_{target},\ \omega_0 - \omega_{target},\ \theta_0 - \theta_{target}\right]^T = 0 \tag{5.2}$$

The selected objective function is generally exploited for a first optimization, where the final aim is to maximize the final mass. The final results typically differ only marginally, but it should be noted that they are not exactly the same. However, the function selected leads to solutions where the development of the control law is more regular and smooth with respect the one obtained for the final mass maximization, granting better performances during the

optimization. These are the reasons behind the habit of choosing the energy integral as objective function.

The specific impulse is assumed greater than a chemical thruster (with a value compatible with the ones associated to electrical thrusters)[1] and an initial mass value is selected considering a large class satellite.
The time-of-flight and the initial control vector are set after some trials. The process behind the first guess generation consists in imposing a simple control law, i.e. same tangential and radial components and zero out-of-plane component for all the stages, and then perform a series of propagation of the dynamic to find a reasonable magnitude of the control and time of flight. The planar transfer obtained as initial guess is quite far from the target orbit due to the significant change of plane aforementioned, and this is the way through which the robustness of HDDP is estimated, in the face of initial guesses far removed from the optimal solution.

$$\begin{aligned} &I_{sp} = 3000\ s \\ &TOF = 7.871\ h \\ &m_0 = 1000\ kg \\ &\mathbf{u}_k = [30,\ 30,\ 0]^T\ N \qquad for\ k = 1, 2, ..., N \end{aligned} \tag{5.3}$$

The specific impulse selected is higher than the ones of real engine that can deliver the level of thrust proposed. However, this value is selected for a first optimization in order to avoid numerical problems associated to a possible excessive decrease in mass, which can take values close to zero as a result of considerable orbital change. For a refinement of the solution with more plausible specific impulses, a continuation scheme can be established in which the obtained solution is used as initial guess for a further optimisation process by setting lower I_{sp}. The results presented in the next subsection are related to the first optimization with the aforementioned I_{sp} because the focus of this study case is related to the robustness of HDDP rather than to obtain data in accord with the true state of the art of the space propulsion technology. However, before to pass to the sensitivity section 5.2, the results obtained by following the continuation scheme will also be presented. The final specific impulse obtained following this procedure is $I_{sp} = 500\ s$, which is compatible with thrust level achieved exploiting a chemical thruster.

5.1.2 Results

The minimum-energy solution obtained consume the 17.55 % of the total initial mass to perform the transfer, incurring in a final mass equal to 824.49 kg. It is important to recall

[1] It shall be underlined that the values of thrust and specific impulses are not retrieved directly from data sheets of some cut-off-the-shelf products

that the small mass consumption is due to the high specific impulse set for the optimization. Setting tolerances $\epsilon_{opt} = 10^{-4}$ and $\epsilon_{feas} = 10^{-5}$, the method converges in 60 iterations. The optimal path presents a norm of the final constraint violation $f = 8.4633 \times 10^{-10}$ and an expected reduction $ER_0 = -1.0305 \times 10^{-8}$. The trend of the constraint violation and the expected reduction along the optimization procedure can be seen in figures 5.2 and 5.3. Moreover, in table 5.2 are reported the values of Lagrange multipliers and penalty parameter related to the initial guess and the optimal solution.
Of particular interest then is the development of the Keplerian parameters of the optimal solution. From an analysis of these trends, in fact, it is easy to glimpse how their behaviour has a correspondence with the physical meaning of the problem. As can be seen by looking at the graphs of the semi-major axis and eccentricity reported in figure 5.4, in a first phase the trajectory leads to a decrease of the two, in order to decrease the timing associated with the transfer and match the orbit in the prescribed TOF. Afterwards, however, the trajectory begins to increase these elements, on the one hand to bring the semi-major axis equal to the target value, and on the other to decrease the control required to perform the change of plane. In fact, as can be seen, the change of inclination is carried out mainly in the second phase of the trajectory, when semi-major axis has already been increased and the eccentricity is high. It is well known from theory on simple impulsive manoeuvres that plane-change manoeuvres are less expensive when performed at large distances from the main attractor. This justifies the high eccentricity. In fact, with the same semi-major axis, the presence of a more elliptical orbit allows for points further away from the focus, where tilt changes can be performed at lower cost.
In order to highlight the considerable difference between initial guess and optimal solution in terms of trajectory, the two are represented graphically in figure 5.5.

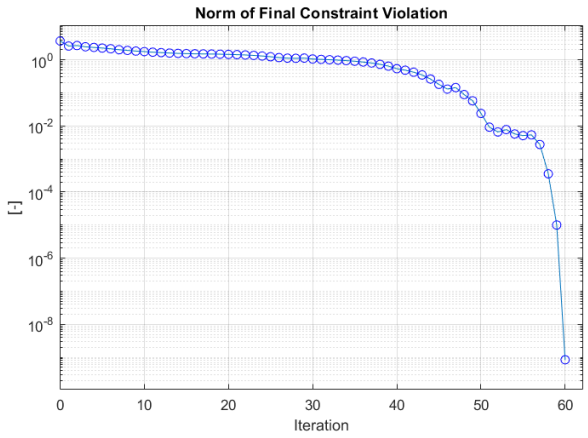


Figure 5.2: Norm of constraint violation for all the iterations

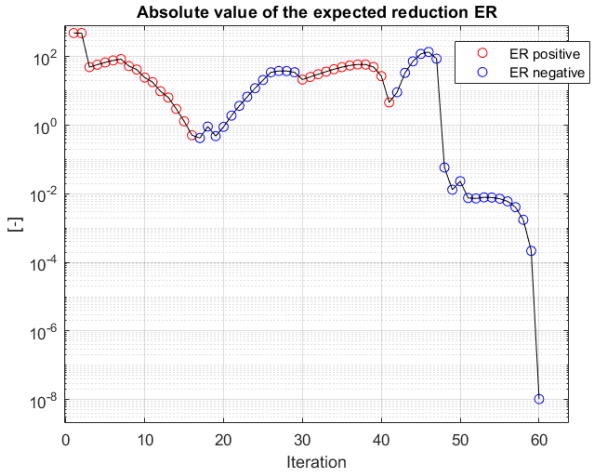


Figure 5.3: Absolute values of the expected reduction ER for all the iterations

	λ_a	λ_e	λ_i	λ_Ω	λ_ω	λ_θ	σ
Initial Guess	0	0	0	0	0	0	1
Optimal Solution	-0.3259	-0.0155	-0.3678	-0.3089	-0.2470	-0.2883	3.375

Table 5.2: Lagrange Multipliers and Penalty Parameter of Initial guess and Final Optimal Solution

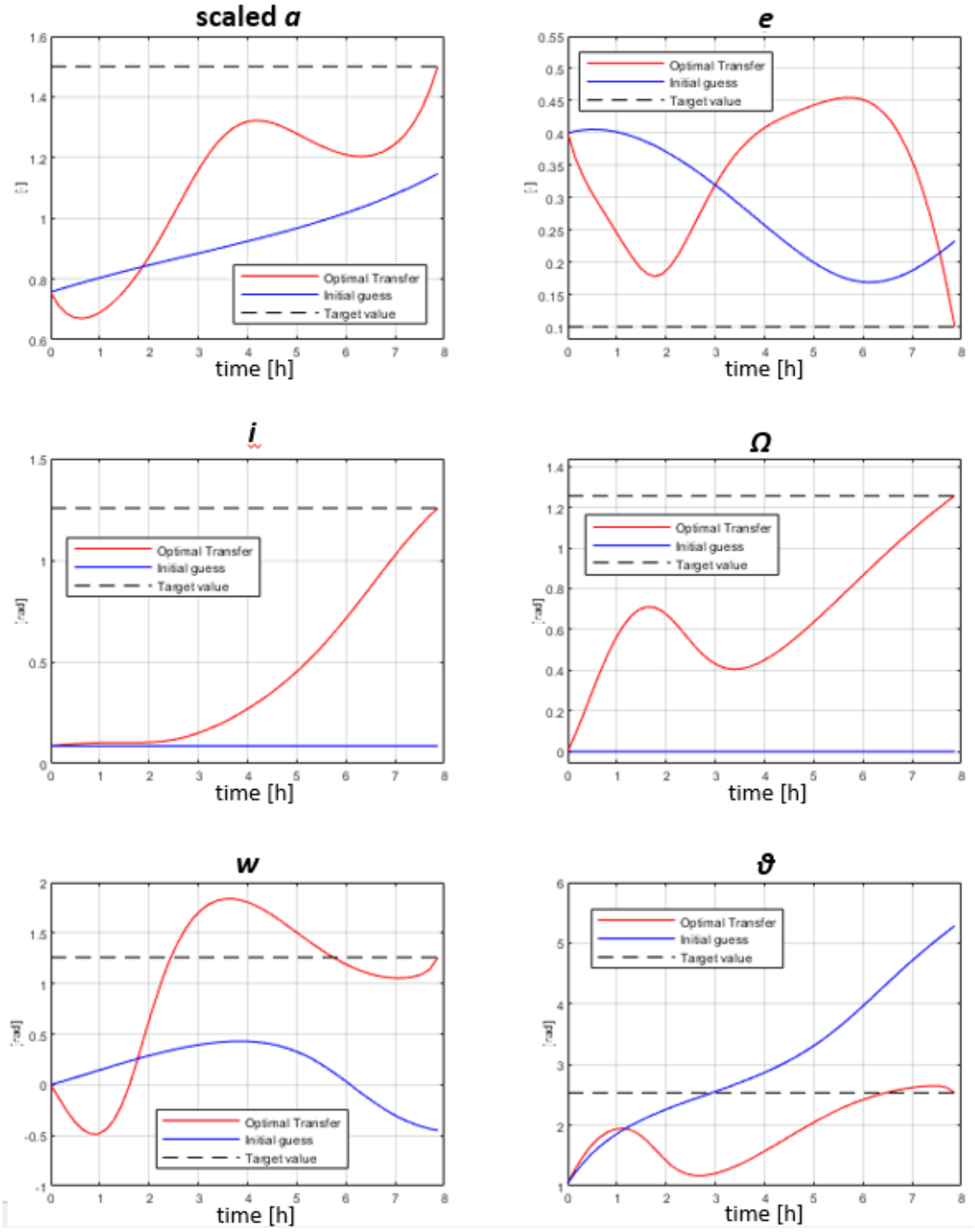


Figure 5.4: Evolution of Keplerian elements in time along the optimal transfer

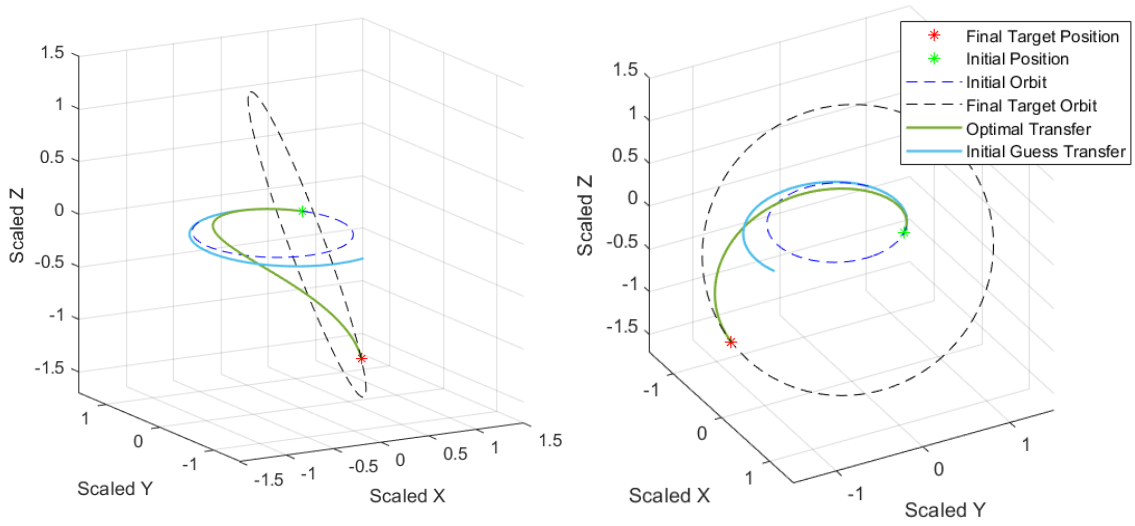


Figure 5.5: Different views of Initial Guess and Optimal Trajectory in scaled 'x,y,z' frame

5.1.3 Specific Impulse Refinement

At this point, from the results obtained for this specific impulse value, a continuation scheme is executed by decreasing the latter first to 800 s and then to 500 s, keeping the same termination tolerances $\epsilon_{opt} = 10^{-4}$ and $\epsilon_{feas} = 10^{-5}$.
At the beginning of each new optimisation, the optimal solution obtained in the previous step is used as the initial guess. At the end of this procedure, the final results obtained presents a final constraint violation $f = 8.4339 \times 10^{-6}$ and an expected reduction $ER_0 = -9.6693 \times 10^{-6}$. The development of the Keplerian elements in time and the trajectory of the optimal transfer are almost identical to the first optimal solution with $I_{sp} = 3000$ s, but the thrust level is lower as a result of the significant reduction in mass due to a lower I_{sp}. Indeed the final optimal transfer consume the 68.39 % of the total initial mass, resulting in a final spacecraft mass equal to 316.05 kg.
The trend of the thrust magnitude and mass consumption related to the final optimal solution are respectively reported in figure 5.6 and figure 5.7.

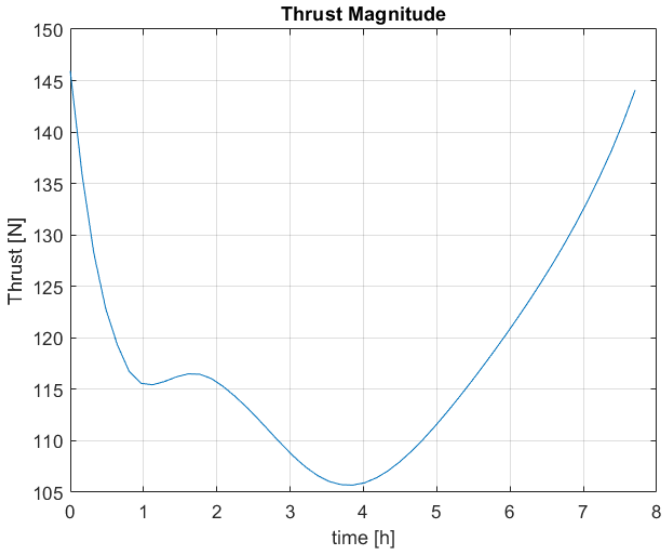


Figure 5.6: Thrust Magnitude associated to the Optimal Trajectory

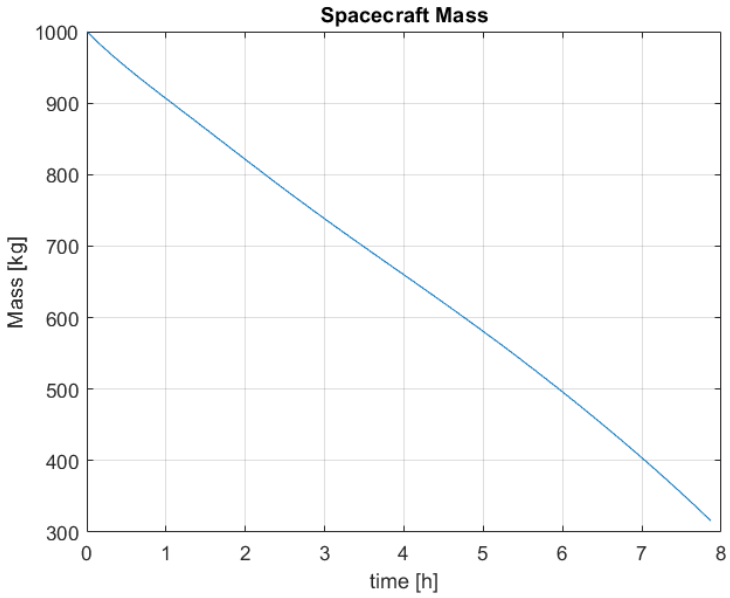


Figure 5.7: Mass of the Spacecraft for the Optimal Trajectory

A last remark should be made. In the study case reported, from the beginning all 6 state variables have been constrained to match the final target. However, it has been noted

during optimization of different initial/final conditions, that sometimes constraining all 6 Keplerian elements from the beginning can generates some difficulties in convergence. Indeed, this is due to the different behaviour of θ and the remaining elements, which have a much less oscillating development than the former. When such a case occurs, it is convenient to constrain the first five orbital parameters in order to obtain a trajectory that is efficiently directed to the target orbit. Subsequently, the solution found can be used as initial guess of a second optimisation process where θ is also constrained. In this way the convergence difficulty seems to be overcome.

5.2 Sensitivity analysis

After deriving the aforementioned optimal solution, it was decided to carry out a study of how the Keplerian elements and final mass vary with time of flight. To achieve this, another continuation scheme was generated, where each optimal solution obtained is used as an initial guess for the problem with different TOF. Thanks to this scheme, different optimal solutions are computed in few iterations due to the initial guess selected, which is already quite close to the optimal condition.
The constraints on the final state are kept identical to those presented in the previous section, as the termination tolerances[2]. The reference TOF is the one that refers to that solution, equal to 7 h 52 min. The analysis was carried out over a window of flight times ranging from a minimum of 5 h 37 min to a maximum of 16 h 07 min, with variations of 45 min between two consecutive solutions, involving a total of 15 trajectories. In the table 5.3 are reported all the major data associated to the converged optimal transfers.

TOF	f [-]	ER_0 [-]	J^* [-]	max thrust [N]	final mass [kg]
5 h 37 min	3.0286×10^{-3}	-1.5718×10^{-4}	0.4807	305.7355	165.959
6 h 22 min	4.1466×10^{-6}	-7.6968×10^{-6}	0.3393	226.3991	219.347
7 h 07 min	5.6969×10^{-7}	-4.2940×10^{-7}	0.2628	182.5557	271.782
7 h 52 min	8.4339×10^{-6}	-9.6693×10^{-6}	0.2093	145.9407	316.055
8 h 37 min	1.1687×10^{-7}	-2.3388×10^{-9}	0.1724	121.4939	350.528
9 h 22 min	7.5005×10^{-7}	-4.2011×10^{-7}	0.1474	107.7695	375.410
10 h 07 min	8.3529×10^{-6}	-2.3622×10^{-6}	0.1306	99.9766	391.178
10 h 52 min	4.6747×10^{-7}	-1.0142×10^{-8}	0.1193	93.9889	399.029
11 h 37 min	1.6009×10^{-6}	-4.5849×10^{-6}	0.1114	89.3096	400.695
12 h 22 min	8.0233×10^{-6}	-3.9113×10^{-6}	0.1060	86.9750	397.848
13 h 07 min	8.3763×10^{-6}	-4.8616×10^{-6}	0.1021	85.1832	391.665
13 h 52 min	3.4305×10^{-6}	-2.8116×10^{-6}	0.0993	83.6534	383.342
14 h 37 min	2.3675×10^{-6}	-1.0015×10^{-6}	0.0973	82.2044	373.775
15 h 22 min	8.3166×10^{-6}	-2.2007×10^{-6}	0.0958	80.7811	363.596
16 h 52 min	3.2778×10^{-7}	-1.0836×10^{-7}	0.0946	79.3189	353.190

Table 5.3: Major data of optimal trajectories with different TOF

[2]Only the solution with the shortest time of flight is stopped before due to proximity to singularity condition

In the next legend are reported in detail all the time of flight of each optimal trajectory and then, the development of each orbital element along time for different solutions are compared. Furthermore, for sake of completeness, also a representation of all the transfers in the scaled x, y, z frame are provided, in figure 5.8.

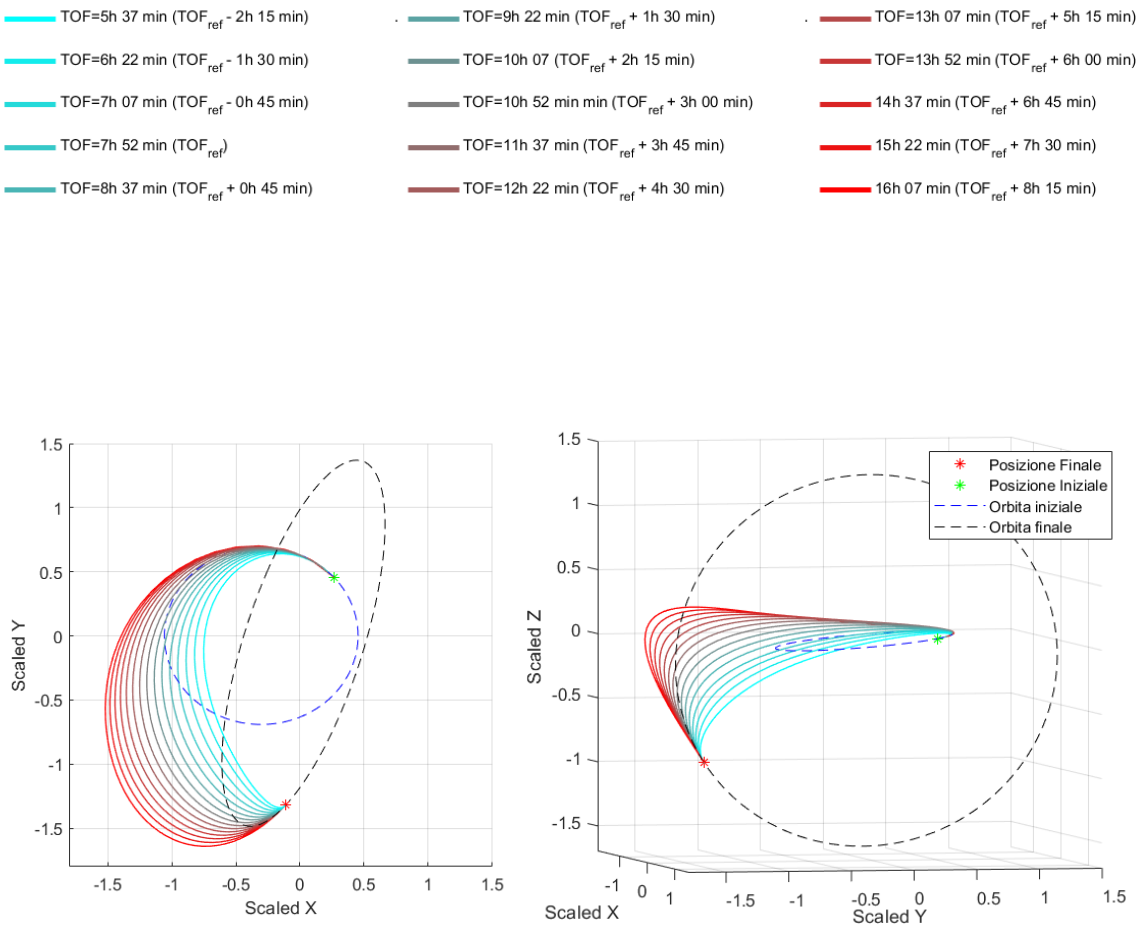


Figure 5.8: Different views of Optimal Trajectories for different TOF in scaled 'x,y,z' frame

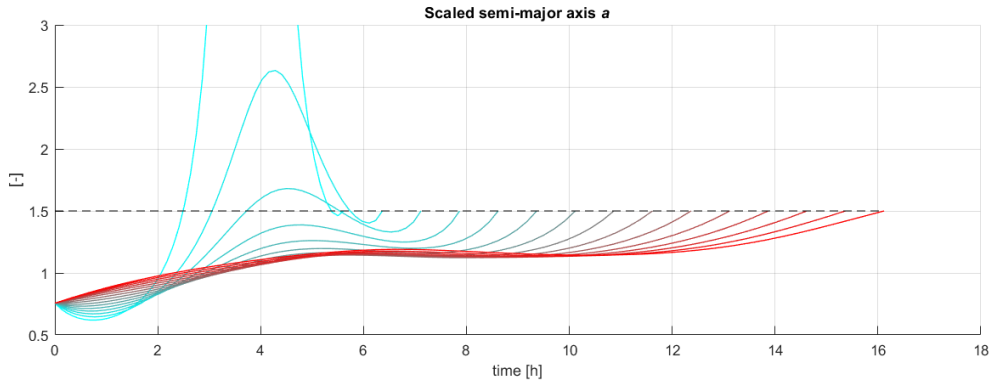


Figure 5.9: Semi-major axis

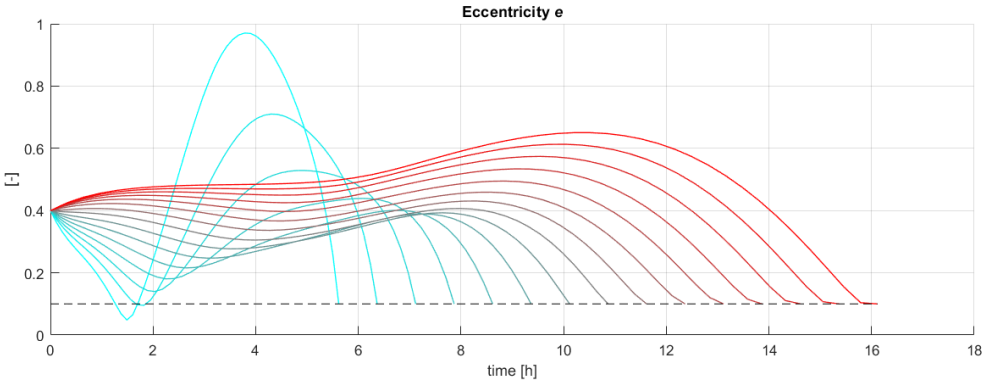


Figure 5.10: Eccentricity

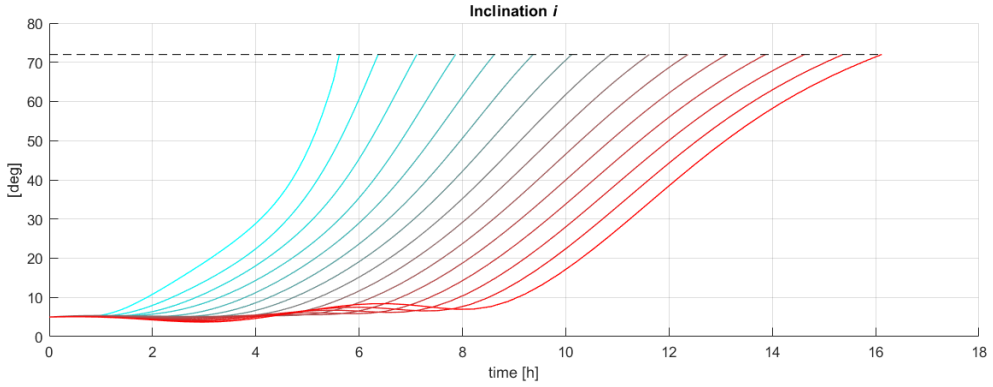


Figure 5.11: Inclination

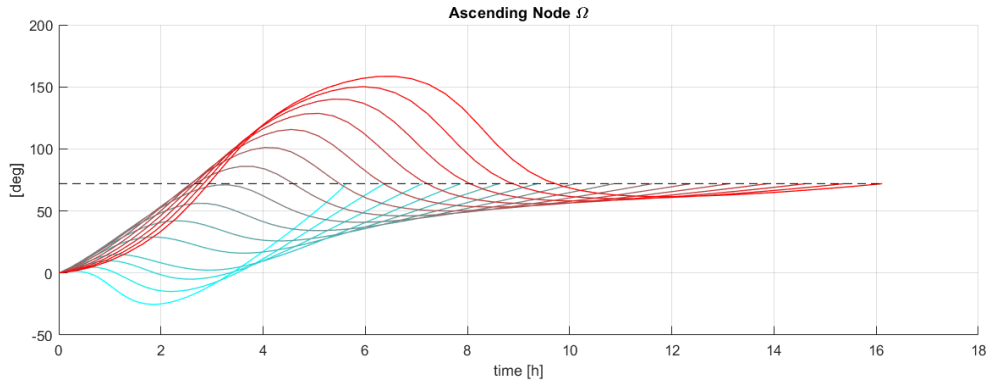


Figure 5.12: Right Ascension of the Ascending Node

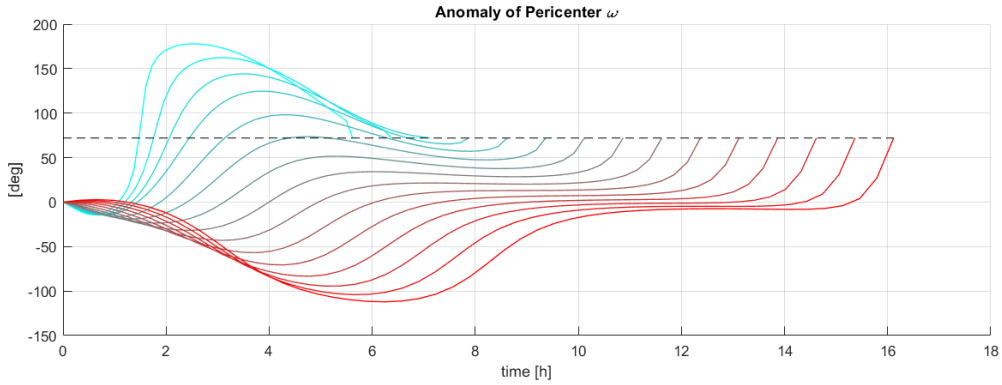


Figure 5.13: Anomaly of the Pericenter

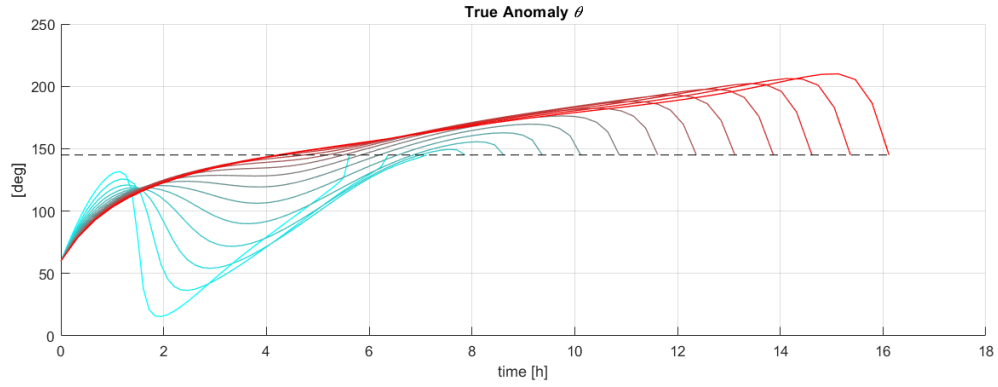
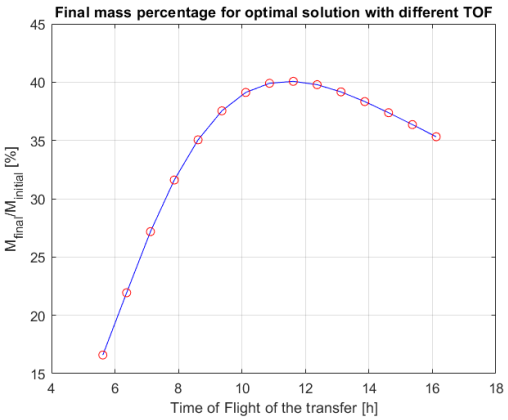


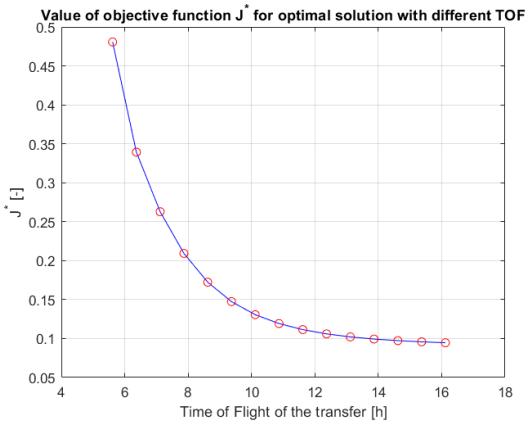
Figure 5.14: True Anomaly

It is interesting to note how the behaviour of the elements changes between solutions. Indeed, when the time of flight is small, there is a remarkable oscillation along the trajectory of both the semi-major axis and the eccentricity, almost reaching two times a singularity condition in the case of minimum TOF (first a very low eccentricity is reached and then a value of the same parameter greater than 0.97). These oscillations, as the time of flight increases, gradually tend to dampen, resulting in an increasingly regular development of the two elements. In fact, the semi-major axis from oscillations that brought it to values above the target one, gradually tends to no longer exceed it along the trajectory. The eccentricity, on the other hand, slowly tends to settle for most of the trajectory in an increasingly flat manner at values close to its initial condition. However, it should be noted that, with a further increase in flight time, the eccentricity again shows an increase in the second part of the transfer before reaching the target value (figures 5.9 and 5.10).
In all solutions, the right ascension has two stationary points, a maximum and a minimum respectively. Both, as TOF increases, take on greater and greater values. Similar trends to Ω are seen for the anomaly of pericenter and the true anomaly, which tend to translate to higher values as the time of flight increases, while the inclination seems to present a similar behaviour in all cases, with its major variation occurring in the final phase of each trajectory.

A further observation to be addressed is related to the final mass obtained for several optimal solutions. In fact, as shown by the figure 5.15a, its trend as a function of TOF is not monotonic, and indeed exhibits a maximum. It hovers at a final mass to initial mass ratio of $\approx 40.07\%$, when $TOF = 11$ h 37 min. It can be deduced, in fact, that beyond a certain value of time of flight, the optimal trajectory needs a greater initial thrust to move further away from the attractor (as shown by the detail in 5.8 and from the detail of figure 5.16) and this manoeuvre seems to be responsible for a greater consumption of propellant. It is interesting to note then that, on the contrary, the trend of the value of the optimal objective function is monotonically decreasing as the TOF increases (figure 5.15b). This, once again, implies that in practice, although they are closely linked, maximisation of the final mass and minimisation of the energy integral are not the same thing.



(a) Percentage Ratio between Final mass to Initial mass for different TOF



(b) J for different TOF*

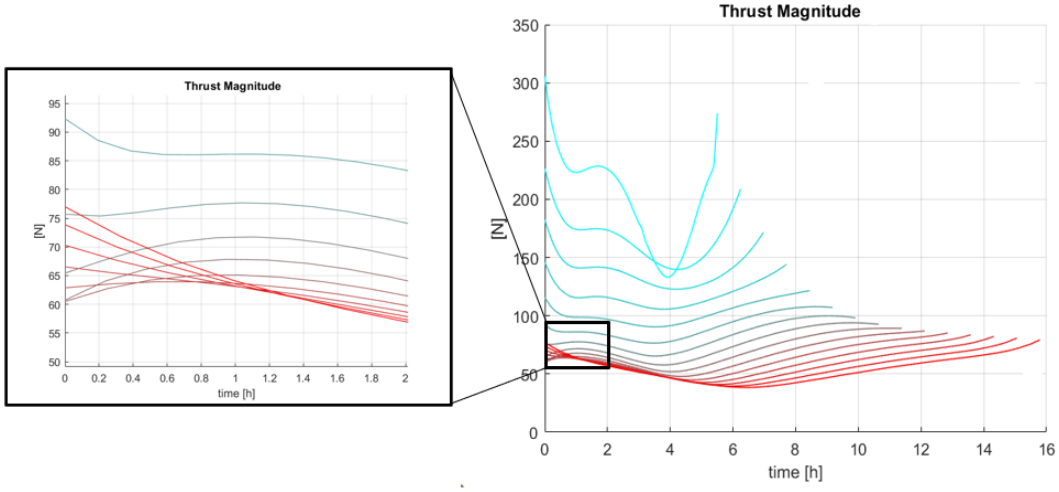


Figure 5.16: Thrust for different optimal solutions

5.3 Interplanetary transfer: Earth-Mars rendez-vous

5.3.1 Description of the study case

A further important application of this algorithm concerns the area of low-thrust transfer trajectories for interplanetary travel, for the various benefits related to the lower consumption brought by modern electric thrusters. In fact, due to the almost total absence of forces at play (very small perturbations when we are in deep space, far from the different planets), the use of low-thrust engines can be an effective strategy for the transfer, all the more so when one considers the gain in mass this solution entails compared to the use of chemical thrusters.
An example often used to test for such a case study is an Earth-Mars rendezvous, which we also decided to propose here. From the point of view of dynamics, due to the presence of orbits with very low eccentricity and inclination, it is convenient to adopt a different set of orbital elements from the previous section, as one is close to singularity conditions. Thus, we chose to use equinoctial elements, which are currently considered the best for two main reasons: from the point of view of dynamics, their evolution does not present any singularities, eliminating the problems associated with the use of the classical Keplerian elements seen previously; secondly, they present an even smoother behaviour than the other orbital elements, and thus allow optimisation to take place with much less difficulty and with fewer iterations.

The departure date selected is 5 June 2023 and arrival date is set to 23 March 2024. The position of the Earth and Mars as function of time are computed by ephemerides model that approximates the ones provided by the NASA JPL website, from which the orbital elements are extrapolated:

	a [AU[3]]	e [-]	i [deg]	Ω [deg]	ω [deg]	θ [deg]
Departure	1.0000002	0.0166993	0	0	103.34	150.77
Arrival	1.5236884	0.0934264	1.85	49.74	286.75	182.21

Table 5.4: Classical Keplerian Orbital Elements associated to Departure/Arrival date

Also in this case a large class satellite is considered and the specific impulse is related to an electric engine. The time of flight is imposed by the departure and arrival date, and the first guess on the control law is retrieved in the same manner as done in section 5.1. At

[3] 1 AU = 149597870707 m

the end of this procedure the data initialized before the optimization are:

$$\begin{aligned} &I_{sp} = 2000\ s \\ &TOF = 292\ days \\ &m_0 = 1000\ kg \\ &\mathbf{u}_k = [100,\ 100,\ 0]^T\ mN \qquad for\ k = 1, 2, ..., N \end{aligned} \tag{5.4}$$

The transfer is subdivided in $N = 70$ discretization stages. Differently from the previous case, in order to show reliability of the algorithm, the objective function selected for this study case consists in the maximization of the final mass m_f.

Furthermore, stage constraints are also introduced here to show how the method deal with such an eventuality. Indeed two constraints on the maximum thrust deliverable are taken into account. The first one is related to the maximum absolute value on the thrust that the engine can provide when it has access to the whole power needed. The other one consider that during the interplanetary travel, if the source of power is provided by solar panels (it is the typical situation, excluding the case when an RTG is used), it decreases as the spacecraft gets far from the Sun. Consequently, the maximum thrust that can be delivered will also decrease as a function of the square of Sun distance, the same trend of the power decrement. Thus, the constraint is determined in order to have $T_{max} = 468.64\ mN$ in correspondence with Mars. They are at the end formulated in the following way:

$$\begin{aligned} &||\mathbf{u}_k|| \leq 700\ mN \\ &||\mathbf{u}_k|| \leq 1300 \left(\frac{r_0}{r}\right)^2\ mN \end{aligned} \tag{5.5}$$

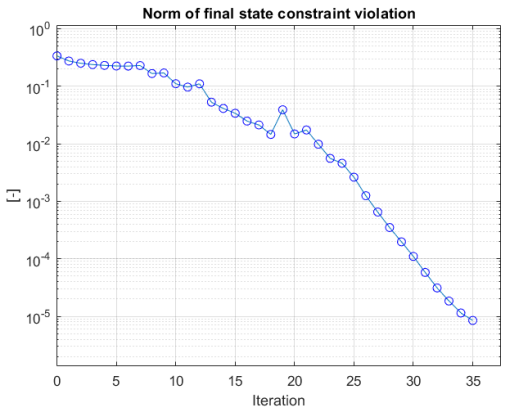
where the distance from the Sun r is computed in the following way $r = \frac{\tilde{a}(1-f^2-g^2)}{1+fcos(L)+gsin(L)}$.
The expression of the final state constraint mirrors what already described in the previous section, the only thing that changes is that it is written in terms of equinoctial target elements.
The tolerances set for the termination are again $\epsilon_{opt} = 10^{-4}$ and $\epsilon_{feas} = 10^{-5}$.

5.3.2 Results

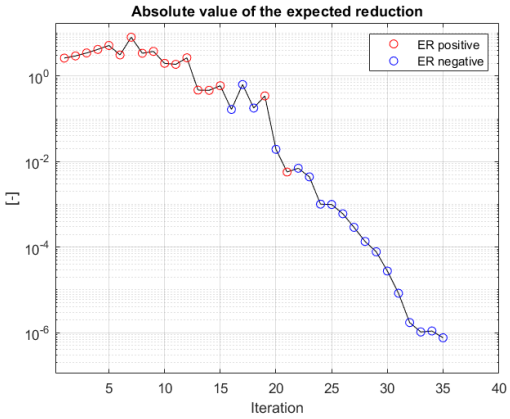
The optimal solution obtained present a final mass equal to 348.82 kg.
With the tolerances selected a final expected reduction $ER_0 = -7.6017 \times 10^{-7}$ and a final state constraint violation $f = 8.4972 \times 10^{-6}$ are obtained. The optimal Lagrange multipliers values and penalty parameter are reported in table 5.5.

λ_a	λ_f	λ_g	λ_h	λ_k	λ_L	σ
-0.9077	-0.4821	1.2162	-0.1398	-0.1590	-0.2764	7.5938

Table 5.5: Values of Lagrange multipliers and Penalty Parameter for the optimal solution



(a) Norm of final state constraint violation for all the iterations



(b) Absolute values of the expected reduction ER for all the iterations

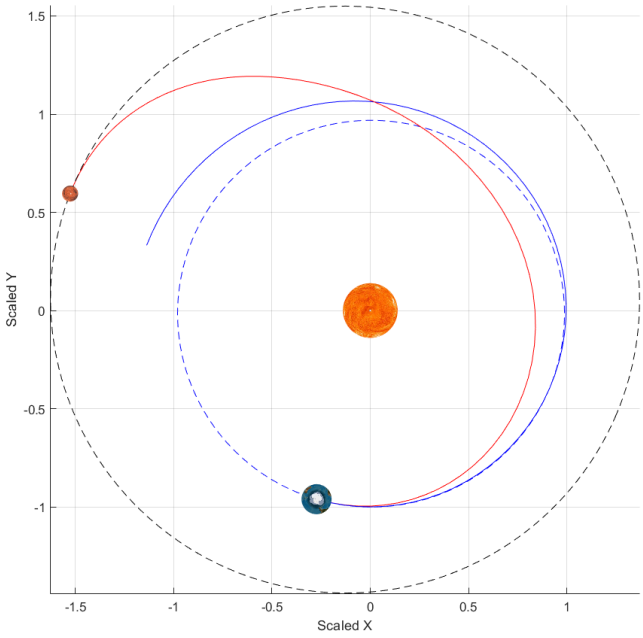


Figure 5.18: Initial Guess and Optimal trajectory represented in the scaled 'x,y,z' reference frame

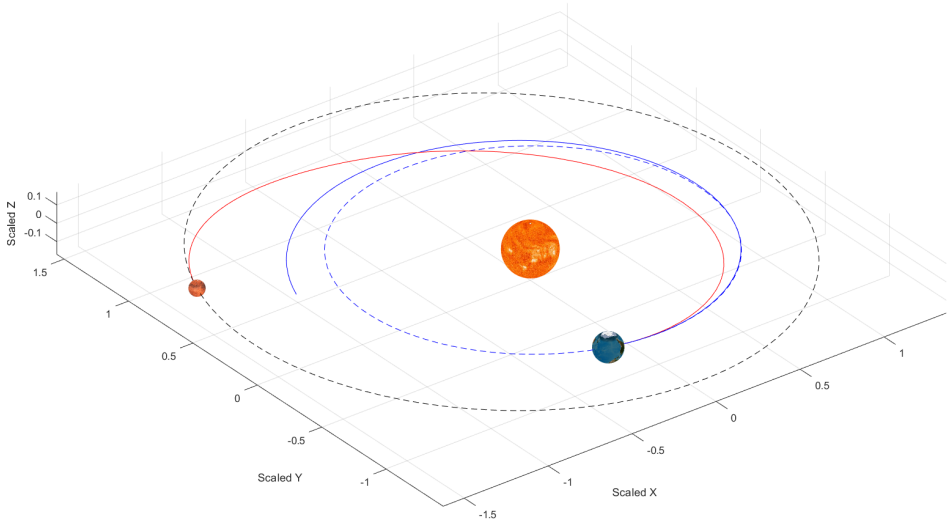


Figure 5.19: Initial Guess and Optimal trajectory represented in the scaled 'x,y,z' reference frame

An interesting feature to observe is the optimal control law obtained. As can be seen from the figure 5.20, the development of the thrust magnitude of the optimal solution is less regular than the one obtained in the previous study case. This is a consequence of the different objective function chosen for the problem under consideration which, as mentioned earlier, tends to generate solutions of this type, similar sometimes to those of a bang-bang control problem. Nevertheless, it is interesting to note that the thrust always remains below the limit values imposed by the two stage constraints introduced, which manifests the ability of the HDDP to handle even this type of restrictions efficaciously.
As can be expected, the introduction of these constraints generates a solution which has an optimal objective function value slightly worsened with respect to the unconstrained case. Indeed, for this specific study case, their lack produces an optimal path with a final mass equal to 413.62 kg, with a control featured by $T_{max} = 1075.97\ mN$.

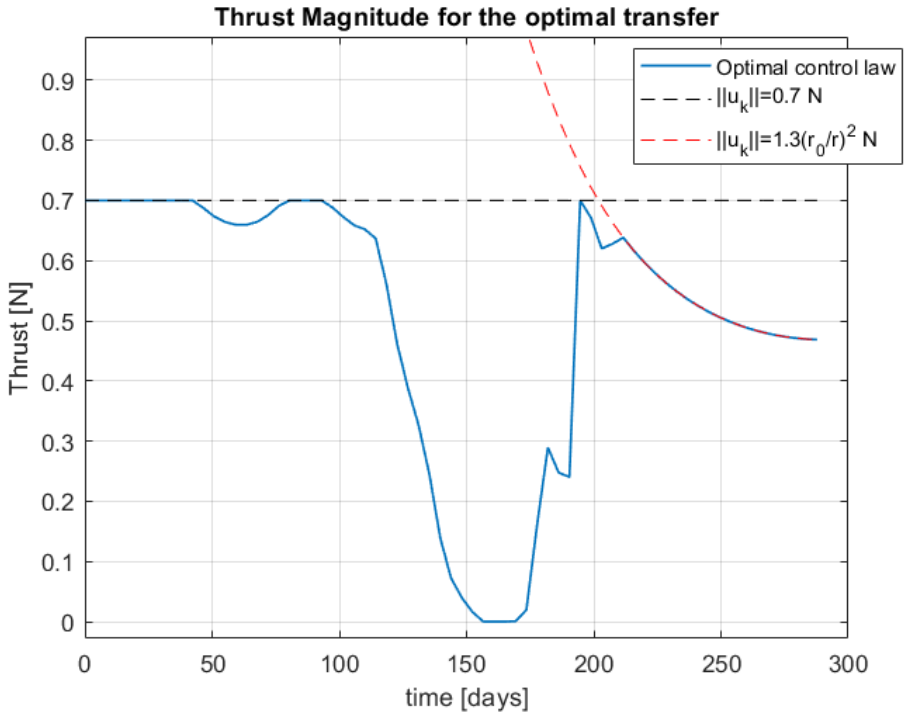


Figure 5.20: Thrust magnitude of the optimal path with stage constraints

Another important aspect to note is the number of iterations in which the solution converges. In this case, in fact, convergence was just achieved in 35 iterations, a symptom of the algorithm's good efficiency. This property is precisely due to the use of orbital elements as a set of variables, instead of the classical Cartesian coordinates. The fact that the former have smoother behaviour than the latter facilitates the algorithm in optimisation, as

manifested by the numerical results. As an example, consider case study number 3 presented in [31]. It consists in an Earth-Mars transfer similar to the one proposed in this thesis work, but which uses the classical set of Cartesian variables as a state vector. The results obtained in terms of final state constraint violation and final trajectory are comparable to the case study presented here, but the optimum is obtained over 172 iterations. Although no real direct comparison can be made between these two problems, it is evident that there is a considerable difference between the number of iterations with which the HDDP reaches convergence when Cartesian coordinates rather than Orbital elements are used as a set of variables.

5.4 Multi-revolution low thrust transfer

The following case study is related to a modern field of application in the area of low-thrust trajectories: multi-revolution transfers in a planetary environment.
Today, in fact, the aforementioned recent implications brought about by the use of electric motors and the increasing miniaturisation capacity of satellites means that low-thrust solutions are increasingly being chosen for transferring from one orbit to another, even when close to a planet. In this context, the limited period associated with orbit combined with low thrust leads to marginal changes in orbital elements after a single revolution. Hence the need to makes a considerable number of revolutions around the attractor before reaching the desired target orbit.

5.4.1 Description of the study case

Following what has just been described, the set of variables selected to describe the problem are once again the equinoctial elements. Three dynamics are used to describe the problem, each of which adopts a different independent variable: the time t, the eccentric anomaly E and the true anomaly θ. An optimization for each dynamic will be performed with the same tolerances, and then a comparison between the solution obtained will be made.
The chosen transfer must connect an inclined eccentric orbit, similar to a GTO, with a geostationary orbit. The keplerian elements of the two are reported in table 5.6.

	a [km]	e [-]	i [deg]	Ω [deg]	ω [deg]
Departure	16578	0.5851	35	0	270
Target	42378	0	0	0	0

Table 5.6: Classical Keplerian Orbital Elements associated to Initial and Target Orbits

The specific impulse is set to 1000 s, and the initial mass of the spacecraft is $m_0 = 1000\ kg$. The transfer is designed in order to perform a number of complete revolution equal to 45 and the discretization stages is set to $N = 300$.
When the problem is formulated as a function of time, the TOF is fixed and the initial guess is generated following the procedure exposed in the previous study cases, whereas when eccentric anomaly or true anomaly are used as independent variable, the number of revolutions is fixed and the magnitude of the control law is selected to generate an initial guess similar to what obtained with time. At the end of this procedures, the final fixed values of the independent variable of each initial guesses are respectively $TOF = 18.3635\ days$, $E_{end} = 286.583\ rad$ and $\theta_{end} = 286.059\ rad$. For all the initial guesses the initial control law is set to $\mathbf{u}_k = [1,\ 1,\ 0]^T$ for all the N stages.
It is important to remark that the solutions can slightly differ because when true or eccentric anomaly are exploited as independent variable, the time of flight is not a fixed parameter.

Vice versa, when time is exploited as independent variable, the final anomaly is not fixed. Therefore, the trajectory can in general modify the "free" parameter in order match the optimal solution.

The optimization consists in the minimization of an objective function similar to the energy integral seen in the first study case. In this case it is represented by the integral on time of the norm of the control, without the squaring. This choice is due to keep the equivalence and avoid some issues in the case of eccentric/true anomaly. Indeed, in these latter cases where the time of flight is not fixed, it can happen that the algorithm try to minimize the objective function favouring the increase of the transfer time over the decrease of the control magnitude, obtaining solutions that have little to do with a first approximation of final mass maximisation. To avoid this situation, the balance between control magnitude and dt will have to be revised by eliminating the squaring on the former. At the end, in order to maintain the equivalence, the objective function in case of eccentric/true anomaly is reformulated as:

$$J_t = \sum_{k=1}^{N} ||\tilde{\mathbf{u}}_k|| \Delta\tilde{t}_k \qquad \Longrightarrow \qquad J_{E,\theta} = \sum_{k=1}^{N} ||\tilde{\mathbf{u}}_k|| \left(\frac{d\tilde{t}}{d\nu}\right)_k \Delta\nu_k \tag{5.6}$$

The final state constraint does not impose any conditions on the true longitude (which is equivalent to not imposing any conditions on the true anomaly) in case of time, neither on time of flight in eccentric/true anomaly cases, in order to evaluate a transfer that fits the target orbit efficiently.
The termination tolerances are set equal to $\epsilon_{opt} = 10^{-4}$ and $\epsilon_{feas} = 10^{-5}$.

5.4.2 Results

At the end of the procedure presented before, the initial guess has the following shape.

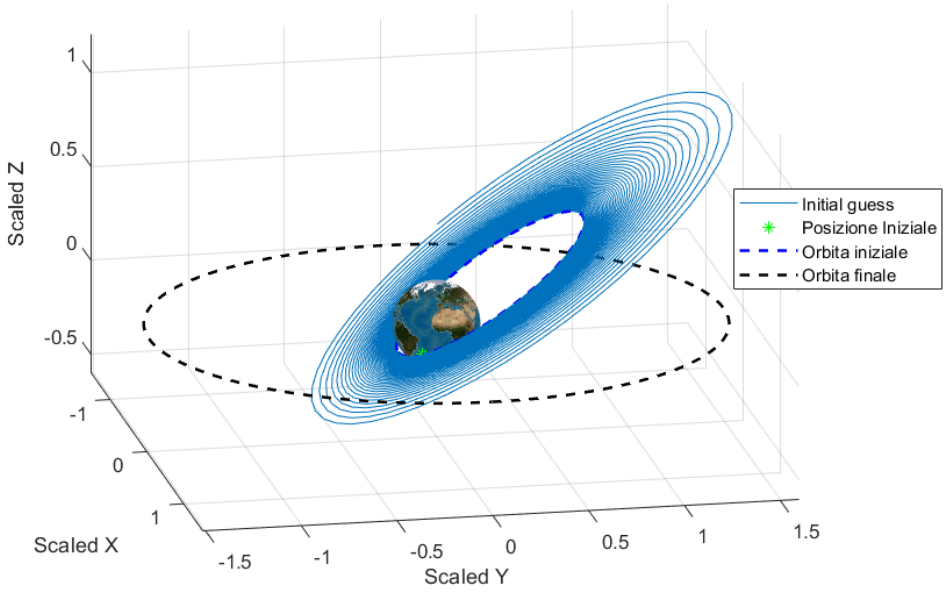


Figure 5.21: Initial Guess of the Multi-revolution Transfer

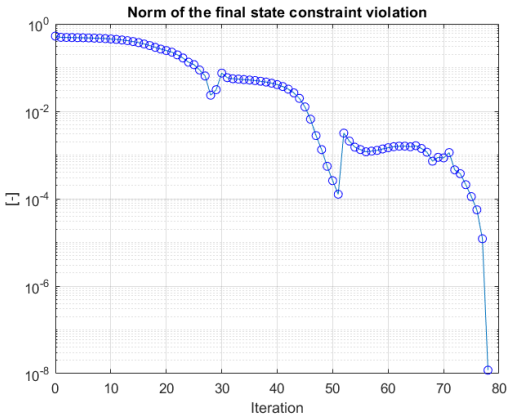
The three optimization produces optimal solutions similar to visualize, but the cases of Eccentric anomaly and True anomaly are associated to a longer time of flight. Because of that, the solution with time is featured by slightly grater values of thrust magnitude required by the transfer.
In terms of final mass, the differences between the three solutions are small (less than 1% of the initial mass); however, the most interesting data for a comparison is the number of iteration required to reach convergence. Indeed, as reported by tab 5.7, the cases which exploits anomalies as independent variable show an efficient behaviour, reaching convergence in just approximately half the iterations needed by the time-dependent case. This demonstrates the convenience of adopting anomalies as independent variable for problem that face a lot of revolutions. The eccentric anomaly case presents itself as the best performing because of the perfect spatial equidistribution of the nodes. However, the use of the true anomaly also proves to be a case with very similar performances, despite the distribution not being perfectly even. This is due, as mentioned earlier, to the fact that the distribution generated follows the same trend associated with the velocity of the dynamics, with a greater accumulation of nodes in the areas where the latter is faster.

Regarding the development between the three solutions, as shown in the figure 5.23, we can see that the development of the optimal solutions expressed in eccentric anomaly and true anomaly is indeed very similar. The difference with the time-dependent case lies in

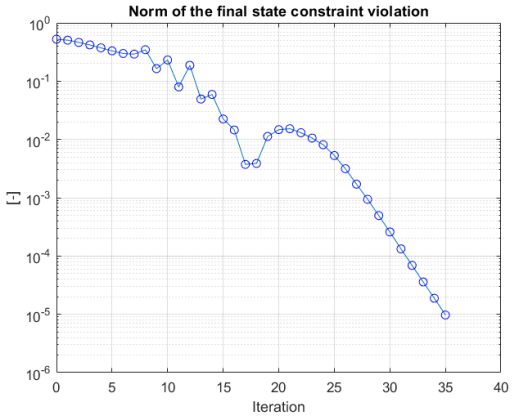
the TOF. In fact, in the first two cases, the solution tends to first increase the semi-major axis (and consequently the associated time of flight) in order to modify the inclination later, while in the time-dependent case, in order to match the target orbit in the prescribed time of flight, the solution is forced to modify the inclination more from the beginning, maintaining a smaller major semi-axis.
To better visualise the differences between the time-dependent case and the cases where an anomaly was chosen as the independent variable, two of the three solutions are shown in Figures 5.24 and 5.25.

	Independent Variable		
	Time	**Eccentric anomaly**	**True anomaly**
ER_0 [-]	-7.8976×10^{-6}	-1.7905×10^{-5}	-3.4589×10^{-5}
f [-]	1.1994×10^{-8}	9.7499×10^{-6}	8.5004×10^{-6}
Final Mass [kg]	659.516	652.274	655.441
Time of flight [days]	18.3635	23.3078	23.4905
Number of revolutions [-]	45	45	45
Final True Anomaly [deg]	246.04	80.37	38.67
J^* [N·s]	0.889246	0.889269	0.846263
Max Thrust [N]	3.403	3.141	3.399
Number of Successful Iterations [-]	**78**	**36**	**42**
Number of Total Iterations [-]	129	84	93

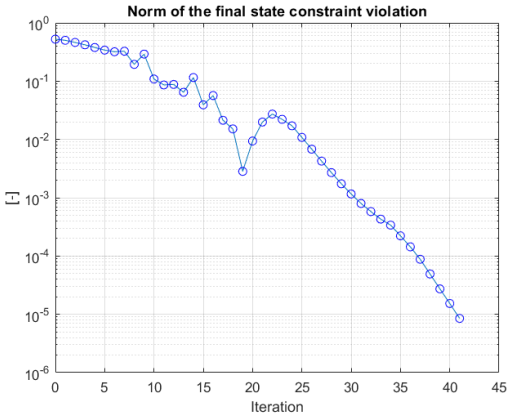
Table 5.7: Most relevant data associated with the 3 different optimal solutions



(a) Time

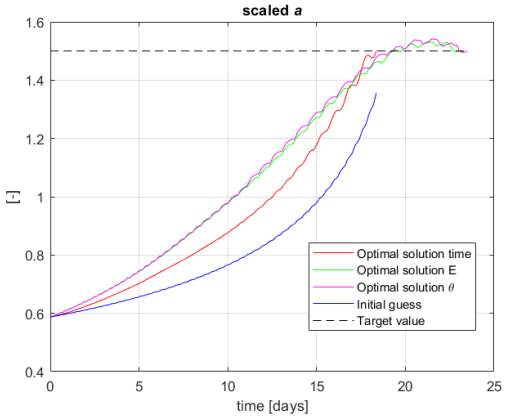


(b) Eccentric Anomaly

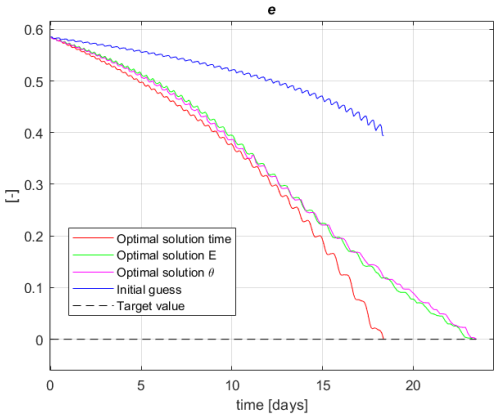


(c) True Anomaly

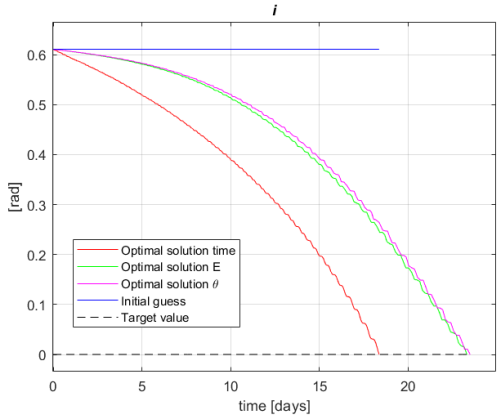
Figure 5.22: Norm of final state constraint violation for all the iteration depending on the independent variable selected



(a) Scaled semi-major axis



(b) Eccentricity



(c) Inclination

Figure 5.23: Evolution in time of different orbital elements of the 3 optimal solutions and the initial guess

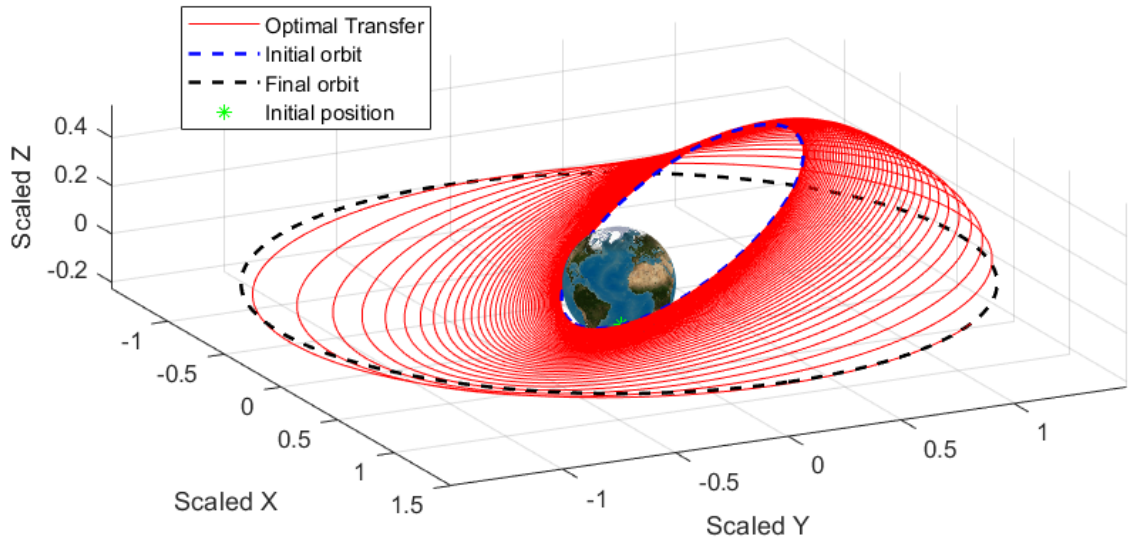


Figure 5.24: Optimal transfer when the independent variable is ***time***

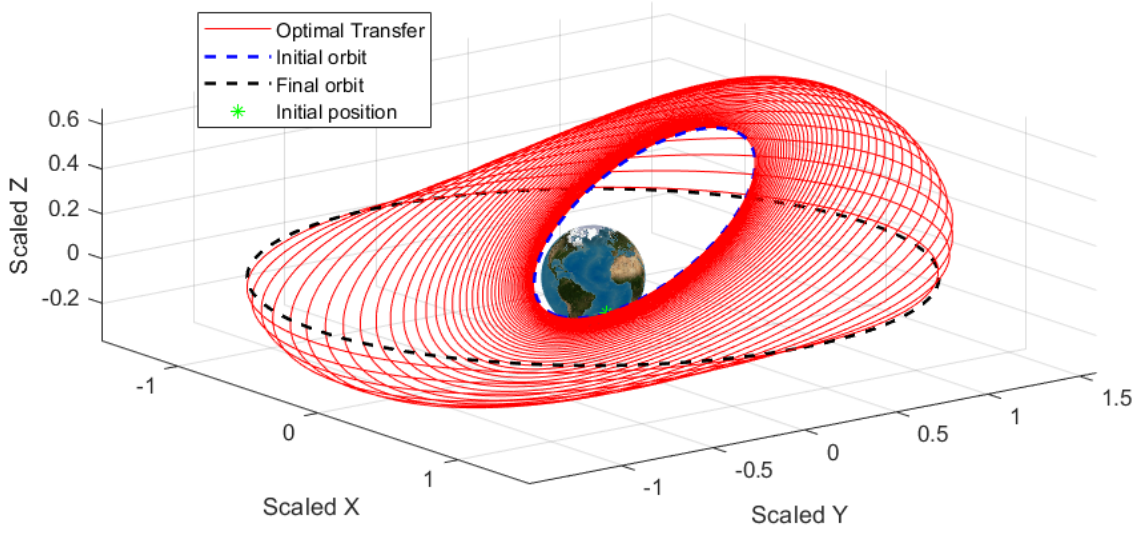


Figure 5.25: Optimal transfer when the independent variable is ***eccentric anomaly***

5.5 Application to a real case: the OneWeb constellation

5.5.1 Description of the study case

As a final case study, it has been decided to apply the method presented to a real mission currently under development. The real application selected consider the orbit raising strategy of the OneWeb satellite's constellation. This constellation aims to build a broadband satellite internet services, consisting in 648 operational satellites of 150 kg each. The plan is to place them in groups of 36 on circular orbits 1200 km above the earth's surface, in 18 different orbital planes. The operational orbits are near polar, at an inclination of 87°.

Figure 5.26: OneWeb Satellite

The first launches of this constellation began in 2019 by Roscomos via the Soyuz-2 rocket. Subsequently, following several financial misadventures of OneWeb Network Access Associates Limited® and the beginning of the Russian-Ukrainian conflict, the company signed a contract with SpaceX® and ISRO in April 2022 to launch the remaining satellites. Thus, the latest developments see the launch of these bodies into orbit by means of the American company's Falcon 9 and the Indian LVM 3. The latter would release them on almost circular orbits of identical inclination and orientation to those planned for actual operation, but at lower altitudes, around 592 km above the earth's surface[4].
The satellites are equipped with Busek-BHT 350 motors, an Hall effect engine endowed with a specific impulse of 1244 s and capable of providing a nominal thrust of 17 mN for a maximum operational time of approximately 6 months[5]. With a variation in power input, it also guarantees a certain level of throttleability, providing a thrust that can be modified from a minimum of 10 mN and to a maximum of 30 mN.

In this context, therefore, the orbit rising strategy consists of moving from a release

[4]the exact data of the released orbit are $a = 592\ km$ and $e = 0.0118$
[5]As listed by the datasheet provided by the Busek official website https://www.busek.com/bht350

orbit at an altitude of 592 km to an operational circular orbit at an altitude of 1200 km. Although the orbital plane of the release orbit is designed to be identical to that of the actual operational orbit, a variation of half-degree has been considered on the initial values of inclination i and right ascension Ω, in order to take into account possible uncertainties due to the imperfect alignment of the launcher's release mechanism.

	a [km]	e [-]	i [deg]	Ω [deg]	ω [deg]
Release Initial Orbit	R_{Terra}[6]+592	0.0118	86.5	0.5	0
Operational Target Orbit	R_{Terra}+1200	0	87	0	0

Table 5.8: Classical Keplerian Orbital Elements associated to Initial and Target Orbits

The large number of revolutions leads to a challenging problem for the algorithm, which requires a large number of discretization stages for a good description of the transfer trajectory. The calculation time becomes considerable in this case, especially due to the computation of the STMs. To limit this, it was decided to parallelize these cycles on the 4 cores available from the PC employed[7]. Furthermore, for the same reason, an attempt was made to limit the number of iterations by using the eccentric anomaly E as the independent variable of the dynamics.

As previously done for the other study case, the initial guess is determined by imposing a simple planar control law (same tangential and radial component, with a magnitude slightly greater than 17 mN, the nominal thrust provided by Busek engine) and some tests were carried out to determine the number of revolutions (until a final semi-major axis is similar to the target one). Once this had been estimated, a reasonable number of discretization nodes N were derived with which the trajectory was subdivided. At the end of this procedure, the main data to be initialized before the optimization are obtained, i.e.:

$$\begin{aligned} &I_{sp} = 1244\ s \\ &m_0 = 150\ kg \\ &N = 2000 \\ &N_{rev} = 400 \\ &\mathbf{u}_k = [14,\ 14,\ 0]^T\ mN \qquad for\ k = 1, 2, ..., N \end{aligned} \tag{5.7}$$

The dynamic is written as a function of the eccentric anomaly as independent variable, and the objective function selected is the energy integral, as already presented in the first case. Due to the small plane change there are not the issues related to the choice of the cost function encountered in 5.4.1, so, differently from the multi-revolution case, the objective

[6] $R_{Terra} = 6378$ km

[7] It is a HP Pavillion 15 with, as processor, an Intel(R) Quad-Core(TM) i7-1065G7 @ 1.3 GHz

function is simply written as:

$$J = \sum_{k=1}^{N} ||\tilde{\mathbf{u}}_k||^2 \Delta\tilde{E}_k \tag{5.8}$$

Due to the remarkable sensitivity to the oscillation of the parameters, and because there is no a prescribed time to reach the exact final position, it has been decided to constraint the first 5 state variables but not the time of flight, as done in the study case presented in section 5.4.1. In addition, because of what has been said earlier about the engine specification, the constraint on thrust is imposed, such that $10\ mN \leq T \leq 30\ mN$ for all the stages.
The termination tolerances are, as always, set to $\epsilon_{opt} = 10^{-4}$ and $\epsilon_{feas} = 10^{-5}$.

5.5.2 Results

The large number of discretization stages implies a long computational time per iteration, mainly due to the numerical integration of first and second order STMs. However, because of the small variation between initial and target values of the Keplerian elements, of the almost planar situation, of the orbital elements exploitation and of the independent variable selected, the method converges in a very limited amount of iterations, $n_{iter} = 3$. This result highlights the potential of using orbital elements combined with HDDP for the optimisation of problems with a large number of revolutions.
The optimal solution presents a final mass equal to 145.875 kg, which is the 97.25% of its initial value, demonstrating a very low propellant consumption required for the transfer. It is featured by an $ER_0 = -5.0175 \times 10^{-14}$ and a final state constraint violation $f = 8.3395 \times 10^{-11}$.
The optimal control law is reported in figure 5.29. As can be seen, even allowing for some misalignment on the release orbit, the equipped engine is able to guarantee the feasibility of the transfer, being the required thrust between a minimum of 14.43 mN and a maximum of 25.29 mN.
Then, in figures 5.27 and 5.28 are reported in detail how semi-major axis and eccentricity change during the transfer.

λ_a	λ_f	λ_g	λ_h	λ_k	σ
-2.2364×10^{-6}	9.6759×10^{-7}	3.9520×10^{-10}	-3.7379×10^{-6}	3.7323×10^{-6}	1×10^{-3}

Table 5.9: Values of Lagrange multipliers and Penalty Parameter for the optimal solution

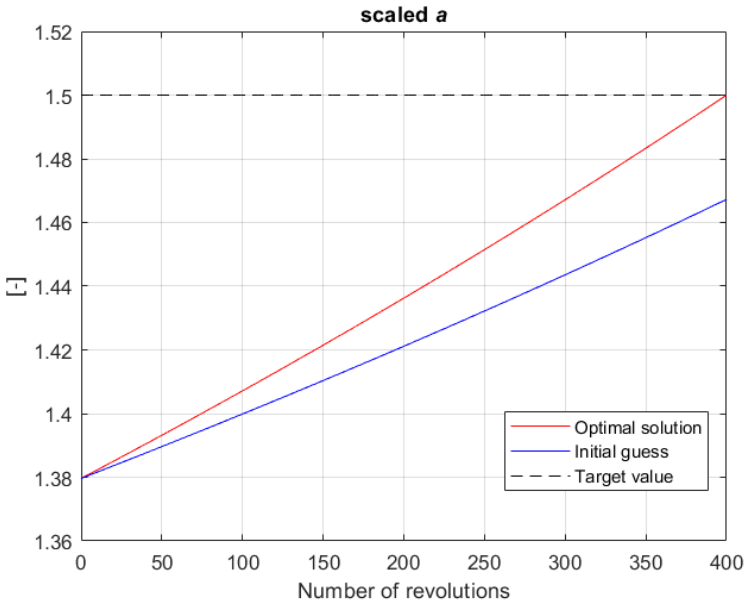


Figure 5.27: Evolution of scaled semi-major axis a of optimal solution and initial guess

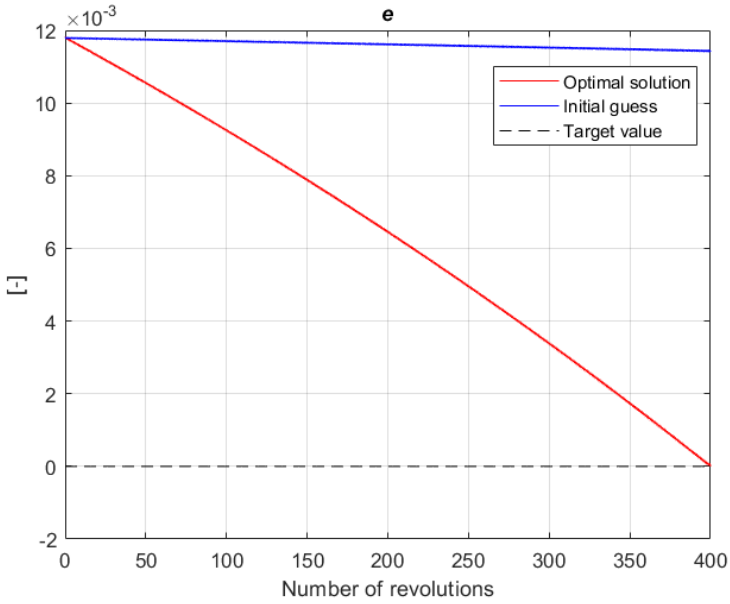


Figure 5.28: Evolution of eccentricity a of optimal solution and initial guess

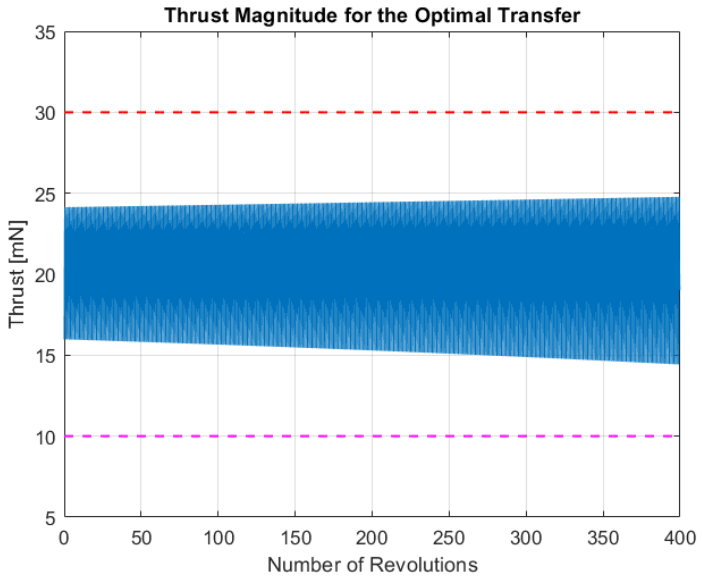


Figure 5.29: Thrust Magnitude of the optimal solution

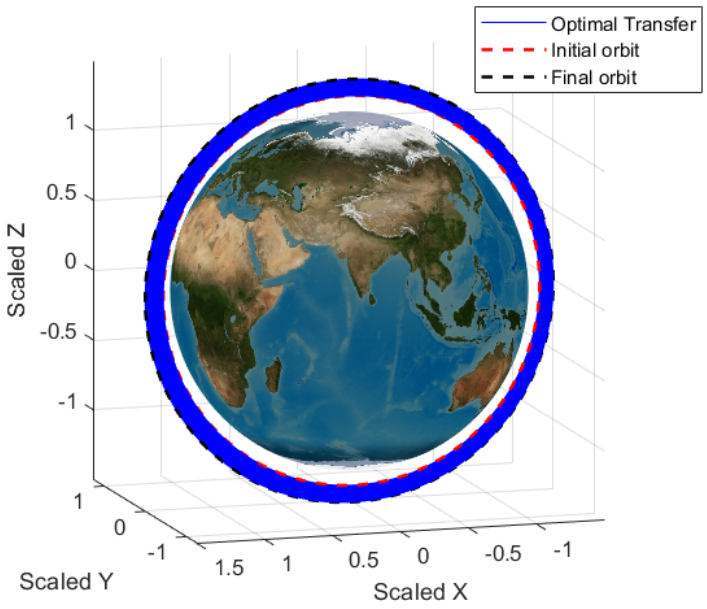


Figure 5.30: Representation of the optimal transfer in the scaled 'x,y,z' reference frame

6 | Conclusions

This thesis deals with optimal problems related to the development of low-thrust and continuous-thrust trajectories in the various fields of application associated with them. As anticipated, these types of transfers are currently an increasingly popular strategy due to the consumption benefits of the new electric thrusters. In fact, they allow significant savings in terms of propellant whenever flight time is not a primary concern.
From an algorithmic point of view, such paths present a number of complications due to their intrinsic nature. In fact, they are associated with a considerable number of discretization nodes (and consequently of decision variables) that pose a general increase in the computational cost of current optimisation methods.
In this context, the application of differential dynamic programming is of particular interest, as it allows results with accuracy comparable to indirect methods and at the same time robustness typical of direct methods at a particularly limited computational cost.
This thesis proposes to combine the use of HDDP algorithm, the last validated upgrade of differential dynamic programming approach, for trajectory optimization with the use of orbital variables. This would allow, as confirmed by numerical results, speeding up convergence in classical problems and increasing the number of revolutions that HDDP can deal with for low-thrust transfers. The latter is a extremely hot-topic for current applications, like orbit servicing, constellation's satellites orbit insertion strategies and multiple target missions. Hence the current relevance of this subject arises.
As confirmed by numerical results, differently from the adoption of classical Cartesian variables as state vector, the use of orbital elements (especially modified equinoctial elements) in HDDP allows the possibility of dealing with multi-revolution problems effectively, with a discretization that require fewer stages. In particular, the introduction of the Sundman transform, makes it possible to further reduce the number of iterations required by the method to achieve convergence, resulting in an increase in the speed of the algorithm whenever it deals with multi-revolution problems. Until now, in fact, only few times optimal multi-revolution transfers have been designed by means of HDDP. In these few cases, the number of revolutions taken into account is however limited. The reformulation of the prob-

lem into orbital elements has considerable advantages for the optimiser, indeed it extends the applicability to cases with a considerable number of revolutions that are impossible to treat exploiting the classical Cartesian variables, as shown by the last study case.
For all these reasons, the thesis aims to introduce the possibility of dealing with low-thrust optimisation problems in all its fields of application in an effective manner thanks this new form, concerning especially multi-revolution cases in the planetary sphere, that prove to be the most challenging problems.
However, the large number of decision variables still poses a significant computational cost in calculating the STMs, a key step in the method. Future developments in this regard will have to focus on reducing the computational cost by adopting methods that replace the typical numerical integration of the latter matrices, such as the exploitation of analytical ones, where this is possible.
The case studies analysed in this work constitute a preliminary optimisation for the realisation of the potential of HDDP and orbital element coupling, laying the foundation for future developments and deeper investigations that can further improve the optimisation method, especially in low-thrust trajectory field.

Bibliography

[1] Bliss G. A. *Lectures on the Calculus of Variations*. Chicago, USA: University of Chicago Press, 1946. ISBN: 978-0226058955.

[2] Petropoulos A.E. "Low-thrust orbit transfers using candidate Lyapunov functions with a mechanism for coasting". In: *AAS/AIAA Astrodynamics Specialist Conference and Exhibit* (Providence, Rhode Island. 2004). URL: https://doi.org/10.2514/6.2004-5089.

[3] Petropoulos A.E. "Simple control laws for low-thrust orbit transfers". In: *AAS/AIAA Astrodynamics Specialist Conference. Pasadena, CA* (2003).

[4] Falb P.L. Athans M.A. *Optimal Control: An Introduction to the Theory and Its Applications*. New York, USA: Dover Publications, 2007. ISBN: 978-0486453286.

[5] Keller H. B. *Numerical Solution of Two Point Boundary Value Problems*. SIAM, 1976. ISBN: 978-0-89871-021-2.

[6] Bertsekas B.P. *Constrained Optimization and Lagrange Multiplier Methods*. San Diego, USA: Academic Press, 1982. ISBN: 978-0120934805.

[7] Shetty C.M. Bazaraa M.S. Sherali H.D. *Nonlinear Programming: Theory and Algorithms*. 3rd edition. Wiley-Interscience, 2006. ISBN: 978-0471486008.

[8] Bulirsch R. Miele A. Stoer J. Well K. Betts J.T. "Trajectory optimization using sparse sequential quadratic programming". In: *Optimal control, International Series of Numerical Mathematics* (1993).

[9] Erb S.O. Betts J.T. "Optimal low-thrust trajectories to the moon". In: *SIAM J. on Applied Dynamical Systems* vol. 2 (2003). DOI: 10.1137/S1111111102409080. URL: https://doi.org/10.1137/S1111111102409080.

[10] Kluever C.A. "Simple guidance scheme for low-thrust orbit transfers". In: *J. of Guidance, Control and Dynamics* vol. 21, n. 6 (1998). DOI: 10.2514/2.4344. URL: https://doi.org/10.2514/2.4344.

[11] Rayman M.D. Fraschetti T.C. Raymond C.A. Russel C.T. "Dawn: a mission in development for exploration of main belt asteroids Vesta and Ceres". In: *Acta Astronautica* vol. 58 (June 2006), pp. 605–616. ISSN: 0094-5765. URL: https://doi.org/10.1016/j.actaastro.2006.01.014.

[12] Nugnes M. Colombo C. "Low-thrust Trajectory Optimisation through Differential Dynamic Programming Method based on Keplerian Orbital Elements". In: *70th International Astronautical Congress (IAC), Washington D.C., United States, paper C1.1.2* (2019).

[13] Radice G Colombo C. Vasile M. "Optimal low-thrust trajectories to asteroids through an algorithm based on differential dynamic programming". In: *Celestial Mechanics and Dynamical Astronomy* vol. (2009).

[14] Toint P.L Conn A.R. Gould N.I.M. *Trust-Region Methods. SIAM, Philadelphia*. Philadelphia, USA: SIAM, 2000. ISBN: 978-0898714609.

[15] Mayne D.Q. "A second-order gradient method for determining optimal control of non-linear discrete time systems". In: *International J. of Control* vol. 3 (1966), pp. 85–95. DOI: 10.1080/00207176608921369. URL: https://www.tandfonline.com/doi/abs/10.1080/00207176608921369.

[16] McReynolds S. Dyer P. *The computation and theory of optimal control*. Vol. vol. 65. Academic Press, 1970. ISBN: 978-0122262500.

[17] Whiffen G.J. "Static/dynamic control for optimizing a useful objective". In: (2002). URL: https://www.freepatentsonline.com/6496741.html.

[18] Jacobson D.H. Gershwin S. "A discrete-time differential dynamic programming algorithm with application to optimal orbit transfer". In: *AIAA J.* vol. 8, n. 9 (1970). DOI: 10.2514/3.5955. URL: https://doi.org/10.2514/3.5955.

[19] Wright M.H. Gill P.E. Murray W. *Practical Optimization*. London, UK: Academic Press, 1981. ISBN: 0.12.283950.1.

[20] W.H. Goodyear. "Completely general closed-form solution for coordinates and partial derivative of the two-body problem". In: *The Astronomical J.* vol. 70 (1965).

[21] Curtis H.D. *Orbital Mechanics for Engineering Students*. 3rd edition. Butterworth-Heinemann, 2013. ISBN: 978-0080977478.

[22] Paris S.W. Hargraves C.R. "Direct Trajectory Optimization using Nonlinear Programming and Collocation". In: *J. of Guidance* vol. 10, n. 4 (Aug. 1987). DOI: 10.2514/3.20223. URL: https://doi.org/10.2514/3.20223.

[23] Aziz J.D Parker J.S. Scheeres D. Englander J.A. "Low-Thrust Many-Revolution Trajectory Optimization via Differential Dynamic Programming and a Sundman Transformation". In: *J. of the Astronautical Sciences* vol. 65 (2018), pp. 205–228. URL: https://doi.org/10.1007/s40295-017-0122-8.

[24] Betts J.T. "Optimal low-thrust orbit transfers with eclipsing". In: *Optimal Control Applications and Methods* vol. 36 (2014), pp. 218–240. DOI: 10.1002/oca.2111.

[25] Mayne D.Q. Jacobson D.H. *Differential Dynamic Programming*. New York, USA: Elsevier, 1970. ISBN: 9780444000705.

[26] Pontryagin L.S. *Mathematical Theory of Optimal Processes*. New York, USA: John Wiley and Sons, 1962. ISBN: 978-0470693810.

[27] Russell R.P. Lantoine G. "A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 1: Theory". In: *J. of Optimization Theory and Applications* vol. 154 (Apr. 2012), pp. 382–417. DOI: 10.1007/s10957-012-0039-0. URL: https://doi.org/10.1007/s10957-012-0039-0.

[28] Russell R.P. Lantoine G. "A Hybrid Differential Dynamic Programming Algorithm for Robust Low-Thrust Optimization". In: *AIAA/AAS Astrodynamic Specialist Conference and Exhibit* (Aug. 2008). DOI: 10.2514/6.2008-6615. URL: https://doi.org/10.2514/6.2008-6615.

[29] Shoemaker C.A. Liao L. "Advantages of Differential Dynamic Programming Over Newton's Method for Discrete-Time Optimal Control Problems". In: (Feb. 1993).

[30] Shoemaker C.A. Liao L. "Convergence in Unconstrained Discrete-Time Differential Dynamic Programming". In: *IEEE Transactions on Automatic Control* vol. 36, n. 6 (1991), pp. 692–706. DOI: 10.1109/9.86943. URL: https://ieeexplore.ieee.org/document/86943.

[31] Maestrini M. "Hybrid Differential Dynamic Programming Algorithm for Low-Thrust Trajectory Design Using Exact High-Order Transition Maps". Master thesis. Politecnico di Milano, 2018.

[32] Nugnes M. "Robust Design of Low-thrust trajectories through Differential Dynamic Programming enhancing the Effects of Orbital Perturbations". Final dissertation for the degree of Doctor of Philosophy in Aerospace Engineering. Supervisor: Camilla Colombo. Politecnico di Milano, Faculty of Industrial Engineering, Department of Aerospace Science and Technologies, 2023.

[33] Yakowitz S. Murray D.H. "Constrained Differential Dynamic Programming with Application to Multi-Reservoir Control". In: *Water Resources Res.* vol. 15 (1979), pp. 1017–1027. DOI: 10.1029/WR015i005p01017. URL: https://doi.org/10.1029/WR015i005p01017.

[34] N. T. T. Program. *Mystic low-thrust trajectory design and visualization software*. URL: https://software.nasa.gov/software/NPO-43666-1.

[35] Bellman R. *Dynamic Programming*. Princeton: Princeton University, 1957. ISBN: 978-0691079516.

[36] Fletcher R. *Practical Methods of Optimization. New York*. New York, USA: Wiley, 2000. ISBN: 9780471915478.

[37] Pellegrini E. Russel R.P. "A multiple-shooting differential dynamic programming algorithm. Part 1: Theory". In: *Acta Astronautica* vol. 170 (2020), pp. 686–700. ISSN: 0094-5765. URL: https://doi.org/10.1016/j.actaastro.2019.12.037.

[38] Pellegrini E. Russel R.P. "A multiple-shooting differential dynamic programming algorithm. Part 2: Applications". In: *Acta Astronautica* vol. 173 (2020), pp. 460–472. ISSN: 0094-5765. URL: https://doi.org/10.1016/j.actaastro.2019.12.038.

[39] Dreyfus S. "Dynamic Programming and calculus of variations". In: *J. of Mathematical Analysis and Applications* vol. 1 (Sept. 1960), pp. 228–239. URL: https://doi.org/10.1016/0022-247X(60)90024-X.

[40] Bulirsch R. Stoer J. *Introduction to Numerical Analysis*. Springer, 2002. ISBN: 978-1441930064.

[41] Edelbaum T. "Propulsion Requirements for controllable satellites". In: *J. of American Rocket Society* vol. 31 (1961), pp. 1079–1089. DOI: 10.2514/8.5723. URL: https://doi.org/10.2514/8.5723.

[42] Edelbaum T. "Theory of maxima and minima". In: *Optimization Techniques, with Applications to Aerospace Systems* vol. 5 (1962), pp. 1–32. URL: https://doi.org/10.1016/S0076-5392(08)62089-5.

[43] Biegler L.T. Wacther A. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* vol. 10 (Mar. 2006).

[44] Alfano S. Wiesel W.E. "Optimal many-revolution orbit transfer". In: *J. of Guidance, Control and Dynamics* vol. 8 (1985), pp. 155–157. DOI: 10.2514/3.19952. URL: https://doi.org/10.2514/3.19952.

A | Trust region subproblem algorithm

In order to solve each iteration of the HDDP, is required to be solved a TRQP mentioned in 3.4.1. This problem can be solved with some different methods, the one selected for this work, due to the small size of control input m, is based on the eigendecomposition of the Hessian $J_{uu,k}$. The fundamental steps of this algorithm are reported here.

Algorithm 1: Adopted Trust region subproblem iterative solving method

Output: $\tilde{J}_{uu,k}$, δu_k
Input: $J_{u,k}$, $J_{uu,k}$, Δ, D

INITIAL SHIFT

Compute eigenvalues $\lambda_1, \lambda_2, ..., \lambda_m$ of $J_{uu,k}$
if all $\lambda_i > 0$ **then**
    $\nu = 0$
else
    $\nu = -2 \min \lambda_i$
end if
$H = J_{uu,k} + \nu I_m$

CHOLESKY DECOMPOSITION OF H:
Find L matrix that $H = LL^T$

FINAL SHIFT

$s = -H^{-1} J_{u,k}$
while $||Ds|| > \Delta$ **do**
    $w = L^{-1} s$
    $\nu = \nu + \left(\frac{||Ds||}{\Delta} - 1 \right) \left(\frac{||Ds||}{||Dw||} \right)^2$
    $H = J_{uu,k} + \nu I_m$
end while

$\tilde{J}_{uu,k} = H$
$\delta u_k = s$

B | Optimisation parameters setting

An important aspect regarding the success of the optimization performed by the HDDP algorithm is the correct setting of the characteristic parameters. These are a lot and each takes on a different meaning, as explained in the section 3.4. The values set to obtain the results seen for the different case studies are provided here. The table B.2 report the data which are kept identical for all the study cases addressed, whereas the table B.1 report the data which are modified for the 4 cases to reach convergence.

	σ_0	Δ_0
Direct transfer	1	1
Earth-Mars Rendezvous	1	0.01
Multi-revolution transfer	0.001	0.001
One Web constellation	0.001	0.001

Table B.1: Some Values of the Parameters set for the optimisation

Parameter Considered	Value
ϵ_{opt}	10^{-4}
ϵ_{feas}	10^{-5}
ϵ_1	0.1
κ	0.25
k_σ	1.5
Δ_{max}	10^3
Δ_{min}	0
D_u	$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{bmatrix}$
D_λ	$\begin{bmatrix} 0.001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix}$

Table B.2: Some Values of the Parameters set for the optimisation

List of Figures

List of Tables