Universidad Politécnica de Madrid

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS AERONÁUTICOS

PROYECTO FINAL DE CARRERA

A Comparative Study of Optimal Control Algorithms using Ideal Frame References

Author: David Morante González

Supervisor: Manuel Sanjurjo Rivo (UC3M)

Tutor: Jesús Peláez Álvarez (UPM)

A thesis submitted in fulfilment of the requirements for the degree of Aeronautics

November 2014

"Many thanks to my supervisor, Dr. Manuel Sanjurjo Rivo, for all his help and continuous dedication, while doing this research. Also for proofreading and support while writing this project."

"Many thanks to my tutor, Dr. Jesús Peláez, for all his help and support since the beginning, providing me always the best information and opportunities"

"My sincerely gratitude to the members of the Space Dynamics Group of the UPM, for his invaluable help to take the first steps in researching, specially to Hodei Urrutxua."

"My deepest appreciation to my family and close friends, for their unselfish help, encouragement and advice."

UNIVERSIDAD POLITÉCNICA DE MADRID

Abstract

Escuela Técnica Superior de Ingenieros Aeronaúticos
Departamento de Física Aplicada a la Ingeniería Aeronaútica

A Comparative Study of Optimal Control Algorithms using Ideal Frame References

David Morante González

The main goal of this dissertation is to implement and compare performances of several trajectory optimization techniques for low-thrust orbit transfer in the restricted two-body problem, using two different formulations of the dynamics. Results are intended to be useful when facing a more complex problem and it would help to choose the appropriate method or algorithm to find the optimal solution.

The selection of flight profiles that yield the greatest performance plays a substantial role in the preliminary design of flight vehicles. Thus, to guarantee the selection of the best vehicle design, it is important to optimize the profile and control law for each configuration early in the design process. In the field of Space Trajectory Propagation, Special Perturbation Methods based on ideal frame concepts such as DROMO or Deprit represent a suitable alternative to describe the dynamics when accuracy, robustness and efficiency is desired. However, it has not been widely used in Optimization Problems.

Nowadays, there exists such a wide variety of direct and indirect optimization techniques that it is not an easy task knowing how to choose which method could be the most efficient option. The main line followed by consits on transcribing the continuous Optimal Control Problem into a finite Nonlinear Programming problem. Different manners to accomplish the transcription such as Single Shooting, Collocation and the recently developed Pseudospectral methods comparing their performances in terms of accuracy, convergence and computational-cost are considered. In addition, advanced numerical gradient based solvers to compute the solution to the NLP problem are applied, and therefore it is evidenced the necessity to properly exploit all their capabilities.

Based on the numerical results of this work, Collocation and Pseudospectral methods present a great performance when Automatic Differentiation is used to provide gradients, and the number of nodes is carefully chosen.

Acknowledgements

Dr. Manuel Sanjurjo Rivo has been the ideal thesis supervisor. His sage advice, insightful criticisms, knowledge and patient encouragement aided the writing of this thesis in innumerable ways.

I would also like to thank Dr. Jesús Peláez whose steadfast support of this project was greatly needed and deeply appreciated.

Contents

List of Figures

List of Tables

Abbreviations

OCP	**O**ptimal **C**ontrol **P**roblem
NLP	**N**on**L**inear **P**rogramming
FD	**F**inite **D**ifferences
CD	**C**omplex **D**ifferentiation
AD	**A**utomatic **D**ifferentiation
IVP	**I**nitial **V**alue **P**roblem
HBVP	**H**amiltonian **B**oundary **V**alue **P**roblem
SQP	**S**equential **Q**uadratic **P**rogramming
IP	**I**nterior **P**oint
SNOPT	**S**parse **N**onlinear **OPT**imizer
IPOPT	**I**nterior **P**oint **OPT**imizer
GPOPS	**G**auss **P**seudospectral **OP**timization **S**oftware
SDG	**S**pace **D**ymacics **G**roup
KKT	**K**arush-**K**uhn-**T**ucker
LG	**L**egendre-**Gauss**
LGR	**L**egendre-**G**auss-**R**adau
LGL	**L**egendre-**G**auss-**L**obatto
HLGL	**H**ermite-**L**egendre-**G**auss-**L**obatto
ODE	**O**rdinary-**D**ifferential-**S**ystem
PMP	**P**ontryagin's **M**inimum **P**rinciple
SP	**S**petial **P**erturbation
BFGS	**B**royden-l **F**lecher- **G**oldfarb-**S**hanno

Symbols

Optimal Control theory

$\dot{(\cdot)}$	Time derivate
$(\cdot)_l$	Lower bound limit
$(\cdot)_u$	Upper bound limit
$(\cdot)_i$	Discrete value at the i-th time
c	Equality constraints in the NLP problem
$\mathcal{C}$	Constraints for the Collocation method
D	Gauss Differentiation matrix
f	Set of dynamical equations
g	Set of path constraints
h	Step size of integrations schemes
$\mathcal{H}$	Hamiltonian of the system
H	Hessian of the Lagrangian
$\mathcal{J}$	Performance index for the Optimal Control Problem
J	Cost function for the Nonlinear optimization problem
$\mathcal{L}$	Lagrangian of the system
L	Lagrange form of the performance index
L_p	Lagrange interpolating polynomials
$\mathcal{M}$	Constraints for the Multiple Shooting method
N	Number of grid points
n_x	Number of state variables
n_u	Number of control variables
n_c	Number of equality constraints
n_h	Number of inequality constraints
n_z	Number of unknown variables in the NLP problem

$n_{\mathfrak{p}}$	Number of static parameters
$\mathfrak{p}$	Set of static parameters
$\boldsymbol{p}$	Search direction
t	Independent variable
t_0	Initial value for the independent variable
t_f	Final value for the independent variable
u	Continuous control
U	Approximated control
$\mathcal{U}$	Set of feasible controls
x	Continuous state
X	Approximated state
$\mathcal{X}$	Set of feasible states
$\hat{x}$	Augmented continuous state/costate
$\hat{X}$	Approximated augmented state/costate
z	Unknown set of variables for the NLP problem
$\mathcal{Z}$	Set of feasible variables for the NLP problem
λ	Continuous costates
Λ	Approximated costates
μ	Lagrange multiplier associated with path constraints
ν	Lagrange multiplier associated with the boundary conditions
ϕ	Boundary conditions
$\hat{\phi}$	Augmented boundary conditions for the states/costates
Φ	Majer form of the cost functional
ω	Gauss weights
τ	Normalized time $\tau \in [-1, 1]$

Special Perturbation Methods

$\hat{(\cdot)}$	Identify dimensional variables
$(\cdot)'$	Identify Deprit variables
$(\cdot)_0$	Value at the initial time
$(\cdot)_c$	Characteristic values
$\dot{(\cdot)}$	Time derivate

$(\cdot)_{\|\|}$	Vector projection on the orbital plane
$(\cdot)_E$	Refers variables to the Earth
$(\cdot)_L$	Refers variables to the moon
$[(\cdot)_x, (\cdot)_y, (\cdot)_z]$	Vector projection on the orbital frame
$\mathcal{O} : [\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}]$	Orbital reference frame
$\mathcal{I} : [\boldsymbol{i}_I, \boldsymbol{j}_I, \boldsymbol{k}_I]$	Inertial reference frame
$\mathcal{H} : [\boldsymbol{i}_H, \boldsymbol{j}_H, \boldsymbol{k}_H]$	Ideal reference frame
a	Semimajor axis of the ellipse
a_p	Especific acceleration due to the perturbing forces
a_f	Effective acceleration
C	Projection of the eccentricity vector along $\boldsymbol{i}_{H'}$
E	Eccentric anomaly reckoned from $\boldsymbol{i}$
$\boldsymbol{e}$	Eccentricity vector
e	Eccentricity
F	Mean Anomaly reckoned from $\boldsymbol{i}_{H'}$
g	Angle between the eccentricity vector and the versos $\boldsymbol{i}_{H'}$
h	Angular momentum
M	Mean anomaly reckoned from $\boldsymbol{i}$
p	Parameter to simplify notation
q	Parameter to simplify notation
Q	Rotation Matrix
$\boldsymbol{r}$	Position vector
r	Module of the position vector
S	Projection of the eccentricity vector along $\boldsymbol{j}_{H'}$
s	Parameter to simply notation
t	Time
t_f	Final time
$[\epsilon_1, \epsilon_2, \epsilon_3]$	Euler parameters
ζ_1	Projection of the eccentricity vector along $\boldsymbol{i}_H$
ζ_2	Projection of the eccentricity vector along $\boldsymbol{j}_H$
ζ_3	Inverse of the angular momentum magnitude
η	Euler parameter

θ	Angle between the position vector and $\boldsymbol{i}_{H'}$
μ	Gravitational constant
ν	True anomaly
σ	Ideal anomaly
τ	Nondimensional time
ϕ	Mean anomaly reckoned from $\boldsymbol{i}_{H'}$
ω	Angular velocity

Dedicated to my family, specially to my mother, who taught me that the best kind of knowledge to have is that which is learned for its own sake. Without their help, support and encouragement this project would not have been possible.

It is also dedicated to my dear university colleagues, who pushed me over the years to improve on a daily basis and whose advices where necessary to fulfil my degree.

A special mention is deserved by Tania de Anta, who showed me that I am able to achieved every goal I set myself. And by Beatriz Bravo, who always told me that even the hardest task can be accomplished step by step.

Chapter 1

Introduction

1.1 Motivation

The selection of flight profiles that yield the greatest performance plays a substantial role in the preliminary design of flight vehicles, since the use of ad-hoc profile or control policies to evaluate competing configurations may inappropriately penalize the performance of one configuration over another. Thus, to guarantee the selection of the best vehicle design, it is important to optimize the profile and control law for each configuration early in the design process.

Early spacecraft used high thrust chemical rocket engines (Ref.[1]) to move from one orbit to another. The thrust from this rockets is normally approximated as impulsive (Ref.[2]), because they produce a large amount of thrust for a relatively short period of time. Recent developments in astronautical engineering have led to the adoption of low-thrust rocket engines for spacecraft (Ref.[3]). Optimizing the orbital transfer for low-thrust engines is significantly more complicated than optimizing transfers for impulsive engines because a continuous control law must be found and long integrations are necessary to determine whether the control law works or not (Ref.[4]).

The fundamental Optimal Control theory was developed in the 1950s and 1960s. Initially, the resolution of Optimal Control Problems turned out to be efficient only for simple or simplified problems. However, with the advances in computing machines, the numerical methods developed, allowed to solve much more complex problems. An splendid compendium of this techniques are summarized in Ref.[5] and Ref.[6]. In the last few decades "*Collocation Pseudospectral Methods*" have been becoming more and more important, because of their ability to solve Optimal Control Problems more efficiently than other existing methods. Concerning this methods, a good review can be found in Ref.[7].

Historically, the optimal low-thrust transfers have been tackled first with indirect and then with direct methods. The former stem from the Pontryagin's maximum principle (Ref.[8]) that uses the calculus of variations, the latter aim at solving the problem via a standard nonlinear programming procedure. Even if it can be demonstrated that both approaches lead to the same result (Ref.[6]), the direct and indirect methods have different advantages and drawbacks, but they require in any case the solution of a complex set of equations: the Euler-Lagrange equations (Ref.[9]) (indirect methods) and the Karush-Kuhn-Tucker equations (Ref.[10]) (direct methods).

In the field of Space Trajectory Propagation, Special Perturbation Methods could represent a suitable option to describe the dynamics of the system when accuracy is desired. The recently developed DROMO propagator (Ref.[11]) has been proved to be regular, robust and efficient. However, they have not been widely used in Optimization Problems.

Nowadays, there exists such a wide variety of optimization techniques that it is not an easy task knowing how to choose which method could be the most efficient to solve the concerning optimization problem.

1.2 Goals

The main goal of this dissertation is to implement and compare performances of several trajectory optimization techniques for low-thrust orbit transfer in the restricted two-body problem, using two different formulations of the dynamics. The results are intended to be useful when facing a more complex problem and it would help to choose the appropriate method or algorithm to find the optimal solution.

Seeking this major goal, certain intermediate goals must be achieved:

1. Exhaustive revision of the literature on optimal control theory and optimal control methods, in particular, those regarding astrodynamic problems in the two-body restricted problem.
2. Revision of the dynamical formulations based on ideal frame references.
    a) Revision of the dynamical formulation using Deprit variables.
    b) Revision of the dynamical formulation using DROMO variables.
3. Comprehensive review of existing techniques and software for trajectory optimization. Elaborate the necessary routines for the optimal control algorithm.
4. Validation of the codes using the analytical solution in a simple case.

5. Application of the codes to a standard problem, comparing performances between the different formulations of the dynamics, and the different optimization techniques.

1.3 Out-line of the document

The paper is organized as follows.

In Chapter (2) we present the Optimal Control Problem, defining the problem and stating the main optimality criteria for its resolution and discussing the differences between Direct and Indirect approaches. In addition, the most widely used computer tools will be presented.

In Chapter (3) we introduce the concept of the ideal frame underlying some Special Perturbation methods. We present two different formulations for high precision propagators, namely Deprit and DROMO, comparing their performance solving a classical orbit propagation problem.

In Chapter (4) the classical problem of the brachistocrone will be solved. The numerical solution will be compared with the existing analytical one, in order to be ascertained of the correct functioning of the code. Both, Single Shooting and Collocation will be implemented using the free-available solvers IPOPT and SNOPT. Moreover, different state-of-art techniques to provide gradients will be also analysed.

A classical orbit transfer trajectory optimization problem will be considered. On the one hand, in Chapter (5) we are intended to maximize the radius of the final orbit. On the other hand, in Chapter (6) we minimize the fuel consumption. For both cases, several numerical tests applying different methods and solvers will be carried out and discussed, outlining the advantages and disadvantages of the techniques considered and trying to decide when a method is more effective than other. Moreover, DROMO formulation is considered as an alternative to the classical variables. The profit of using the high precision propagator will be discussed.

Finally in Chapter(7) some conclusions and future lines of research are drawn.

Chapter 2

Survey of Numerical Methods for Optimal Control Problems

2.1 Introduction

Optimal Control is a subject where it is desired to determine the inputs to a dynamical system that optimize (*i.e.* maximize or minimize) a specified performance index or cost function while satisfying any constraints on the motion of the system. Because of the complexity of most applications, optimal control problems are most often solved numerically. The main goal of this chapter is to summarize the wide variety of methods, based on the surveys wrote by John T.Betts (Ref.[6]) and Anil V.Rao (Ref.[5]). A special interest will be made on a recent class of pseudospectral methods known as *Gauss Pseudospectral Method* based on Ref.[7].

Typically, numerical approaches for solving OCP's fall into two main groups: *Direct Methods* and *Indirect Methods*. On the one hand, in a Direct Method, the state/-control are discretized in some manner in order to transcribe the infinite dimensional problem into a finite optimization problem, which can be solved using "*Non-Linear Programming*" solvers, or also known as NLP solvers, which use advanced mathematical programming techniques (Ref.[10]). On the other hand, the indirect approach uses the "*Calculus of Variations*" (Ref.[9]) to determine the first order optimality conditions, that leads to a "*Hamiltonian Boundary Value Problem*" (HVBP) to determine candidate optimal trajectories called extremals. In Fig.(2.1), an scheme is shown of the two main methods, as a function of the unknown variables in each technique.

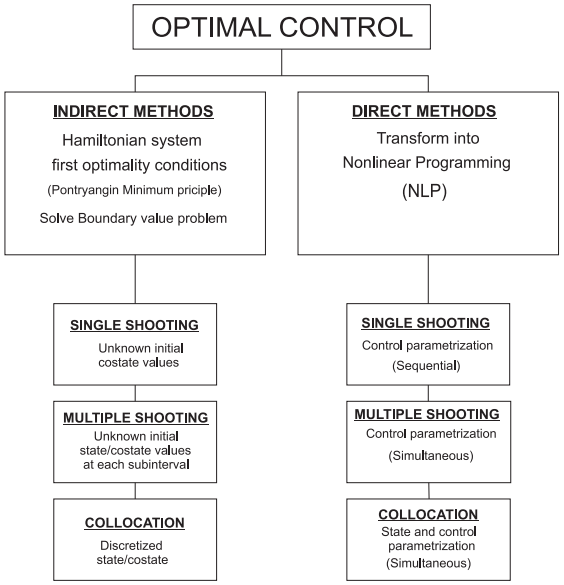


FIGURE 2.1: Different methods for Optimal Control Problems (OCP)

In the following sections, the theory underlying both methods will be presented first, discussing the main advantages and disadvantages of each philosophy, followed by a review of useful computer tools which can be used to solve Optimal Control Problems.

2.2 Methods for solving Optimal Control Problems

2.2.1 Continuous Bolza Problem

Let begin the discussion by formulating the Optimization Problem in a general way. It can be posed formally as follows:

" *Determine the state/trajectory,* $\boldsymbol{x}(t) \in \mathcal{X} \subset \mathbb{R}^{n_x}$*, the control* $\boldsymbol{u}(t) \in \mathcal{U} \subset \mathbb{R}^{n_u}$*, the vector of static parameters* $\mathfrak{p} \in \mathcal{P} \subset \mathbb{R}^{n_\mathfrak{p}}$*, the initial time* $t_0 \in \mathbb{R}$*, and the terminal time* $t_f \in \mathbb{R}$ *(where* $t \in [t_0, t_f]$ *is the independent variable) that optimizes (i.e.minimize or maximize) the following performance index:*"

$$\mathcal{J} = \Phi[\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f; \mathfrak{p}] + \int_{t_0}^{t_f} L[\boldsymbol{x}(t), \boldsymbol{u}(t), t; \mathfrak{p}]dt \tag{2.1}$$

Typically the dynamics of the system are defined by a set of differential equations in a explicit form, which are referred to as the *dynamic constraints*.

$$\dot{\boldsymbol{x}}(t) = f[\boldsymbol{x}(t), \boldsymbol{u}(t), t; \mathfrak{p}] \tag{2.2}$$

and it is subject to the *initial* t_0 and *terminal* t_f *conditions*

$$\boldsymbol{\phi_l} \leq \boldsymbol{\phi}[\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f, ; \mathfrak{p}] \leq \boldsymbol{\phi_u} \tag{2.3}$$

Besides, the solution must satisfy *path constraints*

$$\boldsymbol{g_l} \leq \boldsymbol{g}[\boldsymbol{x}(t), \boldsymbol{u}(t), t; \mathfrak{p}] \leq \boldsymbol{g_u} \tag{2.4}$$

The subscripts l and u refers respectively to the lower and upper bound limits for the constraints.

The objective function (Eq.(2.1)) has been written in terms of quantities evaluated at the end and at the beginning of the trajectory, and also involves an integral. This formulation is called the *Bolza* form. When only the integral is present, the *Lagrange form*, and if the integral does not appear, the *Majer* form. The *Majer* form can be obtained from either *Lagrange* or *Bolza* form by introducing and extra state variable (for proof see *e.g.* Ref.[12]).

Typically the optimal control problem is divided into k-phases and the phases are connected in some meaningful way being subject to linkage constraints. However we will focus on one phase problems. For more information about multiple phase problems see Ref.[6] and Ref.[5].

2.2.2 Indirect Methods

In the indirect approach the set of necessary Optimality Conditions resulting from Pontryagin's minimum principle (Ref.[8]), is directly solved. The *Calculus of Variations* (Ref.[13]) is used to determine the first-order optimality conditions of the OCP given by Eqs.(2.1)-(2.4). These conditions are typically derived using the augmented Hamiltonian $\mathcal{H} \in \mathbb{R}$, defined as

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{u}, t) = L + \boldsymbol{\lambda}(t)^T \boldsymbol{f} - \boldsymbol{\mu}(t)^T \boldsymbol{g} \tag{2.5}$$

where $\boldsymbol{\lambda}(t) \in \mathbb{R}^{n_x}$ is the *costate* or *adjoint* and $\boldsymbol{\mu}(t)$ is the Lagrange multiplier associated with the path constraints. The first-order optimality conditions of the continuous time-problem in the case of a single phase problem with no static parameters, adopt the following form:

The Hamiltonian equations

$$\dot{\boldsymbol{x}} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{\lambda}}, \qquad \dot{\boldsymbol{\lambda}} = -\frac{\partial \mathcal{H}}{\partial x}$$

Pontryagin's Minimum principle

$$\boldsymbol{u}^* = \arg\min \mathcal{H}, \qquad \boldsymbol{u} \in \mathcal{U}$$

Boundary conditions

$$\boldsymbol{\phi}[\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f,] = 0 \tag{2.6}$$

Transversality conditions

$$\boldsymbol{\lambda}(t_0) = -\frac{\partial \Phi}{\partial \boldsymbol{x}(t_0)} + \boldsymbol{\nu}^T \frac{\partial \phi}{\partial \boldsymbol{x}(t_0)}, \qquad \boldsymbol{\lambda}(t_f) = \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)} - \boldsymbol{\nu}^T \frac{\partial \phi}{\partial \boldsymbol{x}(t_f)}$$

The complementary slackness conditions

$$\mathcal{H}(t_0) = \frac{\partial \Phi}{\partial t_0} - \boldsymbol{\nu}^T \frac{\partial \phi}{\partial t_0}, \qquad \mathcal{H}(t_f) = -\frac{\partial \Phi}{\partial t_f} - \boldsymbol{\nu}^T \frac{\partial \phi}{\partial t_f}$$

$$\begin{aligned} &\mu_j(t) = 0, \quad \textit{when} \quad g_j(\boldsymbol{x}, \boldsymbol{u}, t) < 0, && j = 1, ..., n_c \\ &\mu_j(t) \leq 0, \quad \textit{when} \quad g_j(\boldsymbol{x}, \boldsymbol{u}, t) = 0, && j = 1, ..., n_c \end{aligned}$$

where $\mathcal{U}$ is the feasible control set and $\nu \in \mathbb{R}^{n_\phi}$ is the Lagrange multiplier associated with the boundary conditions ϕ.

The Hamiltonian system (Eq.(2.6)) is called a *Hamiltonian Boundary-Value Problem* (HBVP). The optimal solution is then found by choosing the extremal trajectory with the lowest cost. An important issue is that the control is determined by the minimum principle, so that the control can be obtained as a function of the state and costate at each point, explicitly or using a Newton's method (Ref.[14]). Note that, if the Hamiltonian is linear in the control, it turns difficult to find the optimal solution. Specific ways to solve such a kind of problems can be found in Ref.[15].

Given that an indirect method requires solving a two-point boundary value problem, the original OCP is turned into the problem for solving a system of nonlinear equations or "root-finding" of the form:

$$\begin{aligned} c(\boldsymbol{z}) &= 0 \\ h_l \leq h(\boldsymbol{z}) &\leq h_u \end{aligned} \tag{2.7}$$

The primary advantages of indirect methods are their high accuracy in the solution and the assurance that the solution satisfies the first-order optimality conditions. However, indirect methods have several disadvantages, including small radii of convergence, the need to analytically derive the HBVP and a (generally non-intuitive) initial guess for the costate. In addition, the approach is not flexible, since each time a new problem is formulated, a new derivation is required.

The three most common indirect methods are single shooting methods, multiple shooting method and collocation methods.

2.2.2.1 Indirect Single Shooting Method

In a typical single shooting method (Ref.[16]), an initial guess is made of the unknown boundary conditions at one end of the interval. Using this guess, together with the known initial conditions, the Hamiltonian system (Eq.(2.6)) is integrated to the other end. Upon reaching t_f, the terminal conditions obtained are compared to the known terminal conditions (Eq.(2.3)). If the integrated terminal conditions differ from the known terminal conditions by more than a specified tolerance ϵ, the unknown initial conditions are adjusted and the process is repeated again. The process is repeated until the difference is less than the tolerance provided. This process allows to take advantage of the accuracy of the time-marching techniques (Ref.[17]). Note that the control is defined at each point of the integration by the maximum principle (Eq.(2.6)).

Hence, the corresponding problem is to determine the vector of unknown variables $\boldsymbol{z} = \hat{\boldsymbol{x}}(t_0) = [\boldsymbol{x}(t_0), \boldsymbol{\lambda}(t_0)]$, which consists on the initial values at t_0 of the augmented state vector $\hat{\boldsymbol{x}}$ that satisfy:

$$\hat{\boldsymbol{\phi}}[\hat{\boldsymbol{x}}(t_0), t_0, \hat{\boldsymbol{x}}(t_f), t_f,] = 0 \tag{2.8}$$

where $\hat{\boldsymbol{x}}(t_f)$ is the final state of the differential system (Eq.2.6) obtained via a time-marching method and $\hat{\phi}$ are the boundary conditions of the HBVP.

Although the indirect shooting method would seem to be quite straight forward, it suffers from a number of difficulties. Because this method is very sensitive to the initial guess, as it is referred by Bryson in Ref[13] on page 214.

> *The main difficulty with these methods is getting started; i.e., finding a first estimate of the unspecified conditions at one end that produces a solution reasonably close to the specified conditions at the other end. The reason for this peculiar difficulty is the extremal solutions are often very sensitive to small changes in the unspecified boundary conditions...Since the system equations and the Euler-Lagrange equations are coupled together, it is not usual for the numerical integration, with poorly guessed initial conditions, to produce "wild " trajectories in the state space. These trajectories may be so wild that values of $x(t)$ and $\lambda(t)$ exceed the numerical range of the computer!*

Besides, another intrinsic difficulty appears to impose the first-optimality conditions, because it is necessary to analytically differentiate the expressions for $\dot{\boldsymbol{x}}$ and $\boldsymbol{\phi}$, and it may be a daunting task (*e.g.* realistic problems that use complicated aerodynamic, gravitational or propulsion mathematical models). As a consequence, Optimality Conditions can not usually be derived for all possibles combinations or models, which implies a lack of flexibility in indirect methods. However, complex analytical differentiation can be avoided using Automatic Differentiation tools, as it has been implemented in some software like **ADIFOR** or **OCCAL**.

2.2.2.2 Indirect Multiple-Shooting Method

In order to overcome the difficulties of the previous Single Shooting method, the Multiple-Shooting Method has been developed. In this method, the time interval $[t_0, t_f]$ is broken into $N-1$ subintervals. The shooting method is then applied to each subinterval $[t_i, t_{i+1}]$, for $i = 1, \ldots, N-1$ where $t_1 = t_0$ and $t_N = t_f$, with the initial values of the *state* and *adjoints* being unknowns that need to be determined. In order to enforce continuity, the following conditions must be satisfied at the interface of each $i-$subinterval:

$$\mathcal{M} = \hat{\boldsymbol{x}}(t_i^-) - \hat{\boldsymbol{x}}(t_i^+); \tag{2.9}$$

Hence, the respectively "root-finding" problem would consist on determining the variables $\boldsymbol{z}_i = \hat{\boldsymbol{x}}(t_i^+)$ for $i = 1, ..., N-1$ that satisfy the following conditions:

$$\begin{aligned} &\mathcal{M}(\boldsymbol{z}_i) = 0, && for \quad i = 2, ..., N-1 \\ &\hat{\phi}[\boldsymbol{z}_1, t_1, \boldsymbol{z}_N, t_N] = 0 \end{aligned} \tag{2.10}$$

where $\hat{\boldsymbol{x}}(t)$ is the combined state-costate vector defined as $\hat{\boldsymbol{x}}(t) = [\boldsymbol{x}(t), \boldsymbol{\lambda}(t)]$, $\hat{\boldsymbol{x}}(t_i^-)$ is the final state at t_i resulting of integrating Eq.(2.6) in the subinterval $[t_{i-1}, t_i]$; $\hat{\boldsymbol{x}}(t_i^+)$ is the initial state at t_i necessary to performed the integration in the subinterval $[t_i, t_{i+1}]$, $\hat{\phi}$ are the boundary conditions of the Hamiltonian system (Eq.(2.6)).

One obvious result of the multiple shooting approach is an increase of the size of the problem, since new variables and constraints are introduced. In particular, the values of the state and costate at each subinterval represent a new set of variables to optimize. The number of the NLP variables is $n_z = n_{\hat{x}}$ where $n_{\hat{x}}$ is the number of dynamical variables, and $N-1$ is the number of subintervals. Fortunately, the resulting problem is sparse. This sparsity is a direct consequence of the multiple shooting formulation because variables early in the trajectory do not change constraints later in the trajectory. Hence it is important to chose a method that exploits this sparsity.

Perhaps the single most important benefit derived from a multiple shooting formulation is to enhanced robustness, and also the ability to exploit a parallel processor. This code has been implemented in the **BNDSCO** software.

2.2.2.3 Indirect Collocation Method

In an Indirect Collocation Method, the Hamiltonian system (Eq.(2.6)) is integrated via a collocation method instead of a time marching one. The state $\boldsymbol{x}$ and costate $\boldsymbol{\lambda}$ are approximated using piecewise polynomials in the subinterval $[t_i, t_{i+1}]$ for $i = 1, \ldots, N-1$, where $t_1 = t_0$ and $t_N = t_f$. The collocation procedure(Ref.[17]) leads to a root finding problem where the vector of unknowns consists on the approximated values of the augmented state vector $\boldsymbol{z}_i = \hat{\boldsymbol{X}}(t_i) = [\boldsymbol{X}(t_i), \boldsymbol{\Lambda}(t_i)]$ at each grid point t_i. The resultant system must satisfy the collocation equations at the collocation points t_j for $j = 1, \ldots, n_c$, and is then solved using an appropriate root finding technique.

$$\begin{aligned} &\mathcal{C}(\boldsymbol{z}_i) = 0 \qquad\qquad\qquad for \quad i = 1, \ldots, N \\ &\hat{\phi}(\boldsymbol{z}_1, t_1, \boldsymbol{z}_N, t_N) = 0 \end{aligned} \tag{2.11}$$

Where $\mathcal{C}$ represents the defect constraints of the collocation method, and $\hat{\phi}$ the boundary conditions of the Hamiltonian system (Eq.(2.6)).

This approach has been so far less popular, but apparently it has very good perspectives (Ref.[5]). The Matlab® procedure *bvp4c* is an example of effective application of collocation methods based on a finite difference code that implements the 3-stage Lobatto formula. However, its main problem arises when the Jacobian turns singular. This can be avoided by using modified Newton methods (Ref.[10]).

2.2.3 Direct Methods

Direct methods are fundamentally different from indirect methods. The basic idea of direct methods is to transform the infinite Optimal Control Problem into a finite dimensional nonlinear optimization problem. This problems can be solved using Nonlinear Programming techniques (NLP).

An NLP problem takes the following mathematical form:

"*Determine the vector of the unknown decision variables* $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^{n_z}$ *that minimizes the nonlinear cost function* $J(\boldsymbol{z})$"

$$\boxed{\begin{array}{ll} \textit{Minimize} & J(\boldsymbol{z}) \\ \textit{subject to} & c(\boldsymbol{z}) = 0 \\ & h(\boldsymbol{z}) \leq 0 \end{array}} \tag{2.12}$$

where $c(\boldsymbol{z}) \in \mathbb{R}^{n_c}$ represents the equality constraints and $h(\boldsymbol{z}) \in \mathbb{R}^{n_h}$ represents the inequality ones. If the cost function is to be maximized consider $J(\boldsymbol{z})_{max} = -J(\boldsymbol{z})_{min}$ as the objective instead.

The first-order necessary optimality conditions for the NLP are also known as the *Karush-Kuhn-Tucker conditions*(KKT). This conditions can be formulated as follows:

$$c_i(\boldsymbol{z}) = 0, \qquad (i = 1, ..., m) \tag{2.13}$$

$$h_i(\boldsymbol{z}) \leq 0, \qquad (i = 1, ..., p) \tag{2.14}$$

$$\mu_i \geq 0, \qquad (i = 1, ..., p) \tag{2.15}$$

$$\mu_i h_i(\boldsymbol{z}) = 0, \qquad (i = 1, ..., p) \tag{2.16}$$

$$\nabla_z J(\boldsymbol{z}) + \sum_{i=1}^{m} \nu_i \nabla_z c_i(\boldsymbol{z}) + \sum_{i=1}^{p} \mu_i \nabla_z h_i(\boldsymbol{z}) = 0 \tag{2.17}$$

where ν_i and μ_i are known as the Lagrange multipliers due to the equality and inequality constraints respectively. The last equation can be expressed in terms of the Lagrangian of the system as

$$\nabla_z \mathcal{L} = 0 \tag{2.18}$$

where the Lagrangian is given by

$$\mathcal{L} = J(\boldsymbol{z}) + \nu c(\boldsymbol{z}) + \mu h(\boldsymbol{z}) \tag{2.19}$$

Direct methods are more universal and flexible. In a direct method, the state and/or control of the original optimal control problem are approximated in some manner. In the case when only the control is approximated, the method is called a *control parametrization method*. When both the state and control are approximated, the method is called a *state and control parametrization method*. Another distinction can be done between sequential and simultaneous methods. Sequential means that the optimization and simulation processes are carried out separately, while, when simultaneous, they are carried out at the same time. The main advantage of the direct approach consist on the user not having to be concerned with adjoint variables or switching structures.

2.2.3.1 Direct Single Shooting Method

It is the most basic direct method and it is a control parametrization method, where the control is parametrized using a specified functional form, *e.g.*

$$\boldsymbol{u}(t) \approx \boldsymbol{U}(t;\boldsymbol{a}) = \sum_{i=1}^{m} \boldsymbol{a}_i \psi_i(t) \tag{2.20}$$

where $\psi_i(t)$ are known functions of time and $\boldsymbol{a}_i$ are the parameters to be determined from the optimization. The representation of the control can be explicit or implicit. The dynamics is then satisfied by integrating the differential equations with a time-marching method (Ref.[17]). For most applications, a piecewise-linear approximation of the control provide accurate results. Let us divide the subinterval $[t_0, t_f]$ into $N-1$ subintervals $[t_i, t_{i+1}]$, where $i = 0, \ldots, N-1$. The control $\boldsymbol{u}(t)$ will be linearly approximated between $U(t_i)$ and $U(t_{j+i})$ for $t_i < t < t_{i+1}$.

$$\boldsymbol{u}(t) \approx \boldsymbol{U}(t) = \boldsymbol{U}(t_i) + \frac{t - t_i}{t_{i+1} - (t_i)}(\boldsymbol{U}(t_{i+1}) - \boldsymbol{U}(t_i)) \tag{2.21}$$

Then, the optimization problem is to find the vector of unknowns $\boldsymbol{z}_i = \boldsymbol{a}_i, \quad i = 0, \ldots, m$ parameters that describe the control which allow to suit the initial and final constraints. Hence, the NLP that arises from this formulation has the a_i parameters as optimization variables and the problem has the following form:

$$\boxed{\begin{array}{ll} \textit{Minimize} & J(\boldsymbol{z}) \\ \textit{Subject to} & \boldsymbol{\phi}[\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f,] = 0 \end{array}} \tag{2.22}$$

where $\boldsymbol{x}(t_f)$ is the final state of the system Eq.(2.2) which depends on the parameters a_i, ϕ are the boundary conditions(Eq.(2.3)).

The direct shooting method is one of the most widely used methods, mainly on launch vehicle and orbit transfer. Most successful applications have one salient feature in common, namely, the ability to describe the problem in terms of a relatively small number of parameters, but it depends on the control dynamic behaviour. The main difficulty arise in correctly describing the control, and the sensitivity of the variables. Changes early in the trajectory propagate to the end of the trajectory. This is one of the main issues for the use of the direct multiple-shooting method.

2.2.3.2 Direct Multiple-Shooting Method

In a manner similar to that for indirect methods, the interval $[t_0, t_f]$ is divided into $N-1$ subintervals. The aforementioned direct shooting method is then used over each subinterval $[t_i, t_{i+1}]$ with the values of the state at the beginning of each subinterval and the unknown coefficients of the control parametrization given in Eq.(2.20). In order to enforce continuity, the following constraints must be satisfied at the interface of each subinterval.

$$\mathcal{M} = \boldsymbol{x}(t_i^-) - \boldsymbol{x}(t_i^+); \tag{2.23}$$

Hence, the respectively NLP problem would consist on determining the variables $\boldsymbol{z}_i = \boldsymbol{x}(t_i^+)$ for $i = 1, ..., N-1$ that optimize the following problem:

$$\begin{array}{lll} \textit{Minimize} & J(\boldsymbol{z}) & \\ \textit{subject to} & \mathcal{M}(\boldsymbol{z}_i) = 0, & \textit{for} \quad i = 1, ..., N-1 \\ & \phi[\boldsymbol{z}_1, t_0, \boldsymbol{z}_N, t_f] \quad = 0 & \end{array} \tag{2.24}$$

where $\boldsymbol{x}(t_i^-)$ is the final state at t_i resulting of integrating Eq.(2.2) in the subinterval $[t_{i-1}, t_i]$; $\boldsymbol{x}(t_i^+)$ is the initial state at t_i of Eq.(2.2) in the subinterval $[t_i, t_{i+1}]$, ϕ are the boundary conditions Eq.(2.3).

As it can be deduced, the number of variables $n_z = (n_x + n_u)(N-1)$ is augmented because not only the parameters of the control parametrization are to optimize, but the values fo the state at each subinterval. Despite this, this method is an improvement because errors due to the unknown initial conditions are reduced, because the integration is performed in a significant smaller time interval. In addition, the problems is more sparse, because changes early in the trajectory do not affect, the constraints at the end of the interval.

2.2.3.3 Direct Collocation Method

According to Rao (Ref.[5]) *Direct Collocation Methods* are arguably the most powerful methods for solving general Optimal Control Problems. It consists on a state $\boldsymbol{x}$ and control $\boldsymbol{u}$ approximation method, where they both are parametrized using a specified functional form. In this case, the vector of unknown variables of the corresponding NLP consists on the set of values of the state and control at each grid points $\boldsymbol{z}_i = [\boldsymbol{X}(t_i), \boldsymbol{U}(t_i)]$. In this case, the NLP is to optimize the performance index $\mathcal{J}$ (Eq.(2.1)) and the procedure can be stated as follows:

$$\begin{array}{lll} \textit{Minimize} & J(\boldsymbol{z}) & \\ \textit{Subject to} & \mathcal{C}(\boldsymbol{z}_i) = 0 & \textit{for} \quad i = 0, \ldots, N-1 \\ & \phi[\boldsymbol{z}_1, t_0, \boldsymbol{z}_N, t_f] = 0 & \end{array} \tag{2.25}$$

Where $\mathcal{C}$ represents the defect constraints of the collocation method, and ϕ the boundary conditions (Eq.(2.3)).

2.2.4 Pseudospectral Methods

A pseudospectral method is a global form of orthogonal collocation, i.e, the state and control are parametrized using global polynomials, and the differential-algebraic equations are approximated via orthogonal collocation. This method has been efficiently implemented in the software $\mathcal{GPOPS}$ (Ref.[18]) and **DIDO**.

The three most commonly used sets of collocation points are *Legendre-Gauss* (LG), *Legendre-Gauss-Radau* (LGR) and *Legendre-Gauss-Lobatto* (LGL) points. These three sets of points are obtained from the roots of a Legendre polynomial and/or linear combinations of a Legendre polynomial and its derivatives. The LG, LGR, and LGL collocation points lie on the open interval $\tau \in (-1, 1)$, the half open interval $\tau \in [-1, 1)$ or $\tau \in (-1, 1]$, respectively. Let N be the number of collocation points and $P_N(\tau)$ be the Nth-degree Legendre polynomial. The LG points are the roots of $P_N(\tau)$, the LGR points are the roots of $P_{N-1}(\tau) + P_N(\tau)$ and the LGL points are the roots of $\dot{P}_{N-1}$. It is important to note that both the LG and the LGR have non-collocated endpoints. In this section we will focus on *Legendre-Gauss* method, because it is the method implemented in the $\mathcal{GPOPS}$software which will be used in this work. For further information of the remaining methods see Ref.[7].

2.2.4.1 Gauss Pseudospectral Discretization

The direct approach to solving the continuous Bolza Optimal Control Problem of Sec.(2.2.1) is to discretize and transcribe Eqs.(2.1)-(2.4) to a nonlinear programming problem (NLP). The Gauss pseudospectral method, like Legendre and Chebyshev methods, is based on approximating the state and control trajectories using interpolating polynomials. The state is approximated using a basis of $N+1$ Lagrange interpolating polynomials, L_p,

$$\mathbf{x}(\tau) \approx \mathbf{X}(\tau) = \sum_{i=0}^{N} \mathbf{X}(\tau_i) L_{pi}(\tau) \tag{2.26}$$

where $L_{pi}(\tau)$ $(i = 0, \ldots, N)$ are defined as

$$L_{pi}(\tau) = \prod_{j=0, j\neq i}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{2.27}$$

Additionally, the control is approximated using a basis of N Lagrange interpolating polynomials $L^*_{pi}(\tau)$, $(i = 1, \ldots, N)$ as

$$\mathbf{u}(\tau) \approx \mathbf{U}(\tau) = \sum_{i=1}^{N} \mathbf{U}(\tau_i) L^*_{pi}(\tau) \tag{2.28}$$

where

$$L^*_{pi}(\tau) = \prod_{j=1, j\neq i}^{N} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{2.29}$$

It can be seen from Eqs.(2.27) and (2.29) that $L_{pi}(\tau)$ $(i = 0, \ldots, N)$ and $L^*_{pi}(\tau)$ $(i = 1, \ldots, N)$ satisfy the properties

$$L_{pi}(\tau_j) = \begin{cases} 1 & , \quad i = j \\ 0 & , \quad i \neq j \end{cases} \tag{2.30}$$

$$L^*_{pi}(\tau_j) = \begin{cases} 1 & , \quad i = j \\ 0 & , \quad i \neq j \end{cases} \tag{2.31}$$

Differentiating the expression in Eq.(2.26), we obtain

$$\dot{\mathbf{x}}(\tau) \approx \dot{\mathbf{X}}(\tau) = \sum_{i=0}^{N} x(\tau_i) \dot{L}_{pi}(\tau) \tag{2.32}$$

The derivative of each Lagrange polynomial at the LG points can be represented in a differential approximation matrix, $D \in \mathbb{R}^{N \times N+1}$. The elements of the differential

approximation matrix are determined off-line as follows:

$$D_{ki} = \dot{L}_{pi}(\tau_k) = \sum_{l=0}^{N} \frac{\prod_{j=0, j\neq i,l}^{N} (\tau_k - \tau_j)}{\prod_{j=0, j\neq i}^{N} (\tau_i - \tau_j)} \tag{2.33}$$

where $k = 1, \ldots, N$ and $i = 0, \ldots, N$. The dynamic constraint is transcribed into algebraic constraints via the differential approximation matrix as follows:

$$\sum_{i=0}^{N} D_{ki}\mathbf{X}_i - \frac{t_f - t_0}{2}\mathbf{f}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) = \mathbf{0} \quad (k = 1, \ldots, N) \tag{2.34}$$

where $\mathbf{X}_k \equiv \mathbf{X}(\tau_k) \in \mathbb{R}^n$ and $\mathbf{U}_k \equiv \mathbf{U}(\tau_k) \in \mathbb{R}^m$ $(k = 1, \ldots, N)$. Note that the dynamic constraint is collocated only at the LG points and *not* at the boundary points. Additional variables in the discretization are defined as follows: $\mathbf{X}_0 \equiv \mathbf{X}(-1)$, and $\mathbf{X}_f$, where $\mathbf{X}_f$ is defined in terms of $\mathbf{X}_k$, $(k = 0, \ldots, N)$ and $\mathbf{U}(\tau_k)$ $(k = 1, \ldots, N)$ via the Gauss quadrature

$$\mathbf{X}_f \equiv \mathbf{X}_0 + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \mathbf{f}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) \tag{2.35}$$

The continuous cost function of Eq.(2.1) is approximated using a Gauss quadrature as

$$\mathcal{J} = \Phi(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f) + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k L(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) \tag{2.36}$$

where w_k are the Gauss weights. Next, the boundary constraint of Eq.(2.3) is expressed as

$$\boldsymbol{\phi}(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f) = \mathbf{0} \tag{2.37}$$

Furthermore, the path constraint of Eq.(2.4) is evaluated at the LG points as

$$\mathbf{g}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) \leq \mathbf{0} \quad (k = 1, \ldots, N) \tag{2.38}$$

The cost function of Eq.(2.36) and the algebraic constraints of Eqs.(2.34), (2.35), (2.37), and (2.38) define an NLP whose solution is an approximate solution to the continuous Bolza problem. Finally, it is noted that the above discretization can be employed in multiple-phase problems by transcribing the problem in each phase using the above discretization and connecting the phases by *linkage* constraint.

2.2.4.2 Gauss Pseudospectral Discretized KKT Conditions

The first-order optimality conditions (Eq.(2.13)) of the NLP, applying the Gauss Pseudospectral Discretization scheme can be obtained using the augmented cost function or Lagrangian (Eq.(2.19)). The Lagrangian is formed using the Lagrange multipliers $\tilde{\mathbf{\Lambda}}_k \in \mathbb{R}^n$, $\tilde{\boldsymbol{\mu}}_k \in \mathbb{R}^c$, $k = 1, \ldots, N$, $\tilde{\mathbf{\Lambda}}_{\boldsymbol{F}} \in \mathbb{R}^n$, and $\tilde{\boldsymbol{\nu}} \in \mathbb{R}^q$. Using the Gauss discretization scheme it takes the following form:

$$\begin{aligned}\mathcal{L} = \Phi(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f) + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k L(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) - \tilde{\boldsymbol{\nu}}^T \boldsymbol{\phi}(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f) \\ - \sum_{k=1}^{N} \tilde{\boldsymbol{\mu}}_k^T \mathbf{g}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) - \sum_{k=1}^{N} \tilde{\mathbf{\Lambda}}_k^T \left(\sum_{i=0}^{N} D_{ki} \mathbf{X}_i - \frac{t_f - t_0}{2} \mathbf{f}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) \right) \\ - \tilde{\mathbf{\Lambda}}_{\boldsymbol{F}}{}^T \left(\mathbf{X}_f - \mathbf{X}_0 - \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \mathbf{f}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) \right)\end{aligned} \tag{2.39}$$

The KKT conditions are found by setting equal to zero the derivatives of the Lagrangian with respect to $\mathbf{X}_0$, $\mathbf{X}_k$, $\mathbf{X}_f$, $\mathbf{U}_k$, $\tilde{\mathbf{\Lambda}}_k$, $\tilde{\boldsymbol{\mu}}_k$, $\tilde{\mathbf{\Lambda}}_{\boldsymbol{F}}$, $\tilde{\boldsymbol{\nu}}$, t_0, and t_f. The solution to the corresponding NLP must satisfy the following KKT conditions:

$$\sum_{i=0}^{N} \mathbf{X}_i D_{ki} = \frac{t_f - t_0}{2} \mathbf{f}_k \tag{2.40}$$

$$\begin{aligned}\sum_{i=1}^{N} \left(\frac{\tilde{\mathbf{\Lambda}}_i^T}{w_i} + \tilde{\mathbf{\Lambda}}_F^T \right) D_{ki}^{\dagger} + \tilde{\mathbf{\Lambda}}_F^T D_{kN+1}^{\dagger} = \\ \frac{t_f - t_0}{2} \left(-\frac{\partial L_k}{\partial \mathbf{X}_k} - \left(\frac{\tilde{\mathbf{\Lambda}}_k^T}{w_k} + \tilde{\mathbf{\Lambda}}_F^T \right) \frac{\partial \mathbf{f}_k}{\partial \mathbf{X}_k} + \frac{2}{t_f - t_0} \frac{\tilde{\boldsymbol{\mu}}_k^T}{w_k} \frac{\partial \mathbf{g}_k}{\partial \mathbf{X}_k} \right)\end{aligned} \tag{2.41}$$

$$\mathbf{0} = \frac{\partial L_k}{\partial \mathbf{U}_k} + \left(\frac{\tilde{\mathbf{\Lambda}}_k^T}{w_k} + \tilde{\mathbf{\Lambda}}_F^T \right) \frac{\partial \mathbf{f}_k}{\partial \mathbf{U}_k} - \frac{2}{t_f - t_0} \frac{\tilde{\boldsymbol{\mu}}_k^T}{w_k} \frac{\partial \mathbf{g}_k}{\partial \mathbf{U}_k} \tag{2.42}$$

$$\boldsymbol{\phi}(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f) = \mathbf{0} \tag{2.43}$$

$$\tilde{\boldsymbol{\Lambda}}_0^T = -\frac{\partial \Phi}{\partial \mathbf{X}_0} + \tilde{\boldsymbol{\nu}}^T \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}_0} \tag{2.44}$$

$$\tilde{\boldsymbol{\Lambda}}_F^T = \frac{\partial \Phi}{\partial \mathbf{X}_f} - \tilde{\boldsymbol{\nu}}^T \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}_f} \tag{2.45}$$

$$-\frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \frac{\partial \tilde{\mathcal{H}}_k}{\partial t_0} + \frac{1}{2} \sum_{k=1}^{N} w_k \tilde{\mathcal{H}}_k = \frac{\partial \Phi}{\partial t_0} - \tilde{\boldsymbol{\nu}}^T \frac{\partial \boldsymbol{\phi}}{\partial t_0} \tag{2.46}$$

$$\frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \frac{\partial \tilde{\mathcal{H}}_k}{\partial t_f} + \frac{1}{2} \sum_{k=1}^{N} w_k \tilde{\mathcal{H}}_k = -\frac{\partial \Phi}{\partial t_f} + \tilde{\boldsymbol{\nu}}^T \frac{\partial \boldsymbol{\phi}}{\partial t_f} \tag{2.47}$$

$$\mathbf{g}_k \leq \mathbf{0} \tag{2.48}$$

$$\tilde{\mu}_{jk} = 0, \text{ when } \quad g_{jk} < 0 \tag{2.49}$$

$$\tilde{\mu}_{jk} \leq 0, \text{ when } \quad g_{jk} = 0 \tag{2.50}$$

$$\mathbf{X}_f = \mathbf{X}_0 + \frac{(t_f - t_0)}{2} \sum_{k=1}^{N} w_k \mathbf{f}_k \tag{2.51}$$

$$\tilde{\boldsymbol{\Lambda}}_F = \tilde{\boldsymbol{\Lambda}}_0 + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \left(-\frac{\partial g_k}{\partial \mathbf{X}_k} - \left(\frac{\tilde{\boldsymbol{\Lambda}}_k^T}{w_k} + \tilde{\boldsymbol{\Lambda}}_F^T \right) \frac{\partial \mathbf{f}_k}{\partial \mathbf{X}_k} + \frac{2}{t_f - t_0} \frac{\tilde{\boldsymbol{\mu}}_k^T}{w_k} \frac{\partial \mathbf{g}_k}{\partial \mathbf{X}_k} \right) \tag{2.52}$$

where the shorthand notation $L_k \equiv L(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f)$, $\mathbf{f}_k \equiv \mathbf{f}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f)$, $\mathcal{H}_k \equiv \mathcal{H}(\mathbf{X}_k, \boldsymbol{\Lambda}_k, \boldsymbol{\mu}_k, \mathbf{U}_k, \tau_k; t_0, t_f)$, and $g_{jk} \equiv g_j(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f)$ is used. Note that the augmented Hamiltonian, $\tilde{\mathcal{H}}_k$, is defined as

$$\tilde{\mathcal{H}}_k \equiv L_k + \left(\frac{\tilde{\boldsymbol{\Lambda}}_k^T}{w_k} + \tilde{\boldsymbol{\Lambda}}_F^T \right) \mathbf{f}_k - \frac{2}{t_f - t_0} \frac{\tilde{\boldsymbol{\mu}}_k^T}{w_k} \mathbf{g}_k \tag{2.53}$$

and $\tilde{\boldsymbol{\Lambda}}_0$ is defined as

$$\tilde{\boldsymbol{\Lambda}}_0^T \equiv -\frac{\partial \Phi}{\partial \mathbf{X}_0} + \tilde{\boldsymbol{\nu}}^T \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}_0} \tag{2.54}$$

2.2.4.3 Gauss Pseudospectral Discretized Necessary Conditions

In order to discretize the variational conditions of Section (2.2.2) using the Gauss pseudospectral discretization, it is necessary to form an appropriate approximation for the costate. In this method, the costate, $\lambda(\tau)$, is approximated as follows:

$$\lambda(\tau) \approx \Lambda(\tau) = \sum_{i=1}^{N+1} \lambda(\tau_i) L_i^{\dagger}(\tau) \tag{2.55}$$

where $L_i^{\dagger}(\tau)$ $(i = 1, \ldots, N+1)$ are defined as

$$L_i^{\dagger}(\tau) = \prod_{j=1, j\neq i}^{N+1} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{2.56}$$

It is emphasized that the costate approximation is *different* from the state approximation. In particular, the basis of $N+1$ Lagrange interpolating polynomials $L_i^{\dagger}(\tau)$ $(i = 1, \ldots, N+1)$ includes the costate at the *final* time (as opposed to the initial time which is used in the state approximation). This (non-intuitive) costate approximation is necessary in order to provide a complete mapping between the KKT conditions and the variational conditions.

Using the costate approximation of Eq.(2.55), The first-order necessary conditions of the continuous Bolza problem in Eq.(2.6) are discretized as follows. First, the state and control are approximated using Eqs.(2.26) and (2.28), respectively. Next, the costate is approximated using the basis of $N+1$ Lagrange interpolating polynomials as defined in Eq.(2.55). The continuous-time first-order optimality conditions of Eq.(2.6) are discretized using the variables $\mathbf{X}_0 \equiv \mathbf{X}(-1)$, $\mathbf{X}_k \equiv \mathbf{X}(\tau_k) \in \mathbb{R}^n$, and $\mathbf{X}_f \equiv \mathbf{X}(1)$ for the state, $\mathbf{U}_k \equiv \mathbf{U}(\tau_k) \in \mathbb{R}^m$ for the control, $\boldsymbol{\Lambda}_0 \equiv \boldsymbol{\Lambda}(-1)$, $\boldsymbol{\Lambda}_k \equiv \boldsymbol{\Lambda}(\tau_k) \in \mathbb{R}^n$, and $\boldsymbol{\Lambda}_f \equiv \boldsymbol{\Lambda}(1)$ for the costate, and $\boldsymbol{\mu}_k \equiv \boldsymbol{\mu}(\tau_k) \in \mathbb{R}^c$, for the Lagrange multiplier associated with the path constraints at the LG points $k = 1, \ldots, N$. The other unknown variables in the problem are the initial and final times, $t_0 \in \mathbb{R}$, $t_f \in \mathbb{R}$, and the Lagrange multiplier, $\boldsymbol{\nu} \in \mathbb{R}^q$. The total number of variables is then given as $(2n + m + c)N + 4n + q + 2$. These variables are used to discretize the continuous necessary conditions of Eq.(2.6) via the Gauss pseudospectral discretization. Note that the derivative of the state is approximated using Lagrange polynomials based on $N+1$ points consisting of the N LG points and the initial time, τ_0, while the derivative of the costate is approximated using Lagrange polynomials based on $N+1$ points consisting of the N LG points and the final time, τ_f. The resulting algebraic equations that approximate the continuous

necessary conditions at the LG points are given as

$$\sum_{i=0}^{N} \mathbf{X}_i D_{ki} = \frac{t_f - t_0}{2} \mathbf{f}_k \tag{2.57}$$

$$\sum_{i=1}^{N} \mathbf{\Lambda}_i D^{\dagger}_{ki} + \mathbf{\Lambda}_f D^{\dagger}_{kN+1} = \frac{t_f - t_0}{2} \left(-\frac{\partial L_k}{\partial \mathbf{X}_k} - \mathbf{\Lambda}_k^T \frac{\partial \mathbf{f}_k}{\partial \mathbf{X}_k} + \boldsymbol{\mu}_k^T \frac{\partial \mathbf{g}_k}{\partial \mathbf{X}_k} \right) \tag{2.58}$$

$$\mathbf{0} = \frac{\partial L_k}{\partial \mathbf{U}_k} + \mathbf{\Lambda}_k^T \frac{\partial \mathbf{f}_k}{\partial \mathbf{U}_k} - \boldsymbol{\mu}_k^T \frac{\partial \mathbf{g}_k}{\partial \mathbf{U}_k} \tag{2.59}$$

$$\boldsymbol{\phi}(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f) = \mathbf{0} \tag{2.60}$$

$$\mathbf{\Lambda}_0 = -\frac{\partial \Phi}{\partial \mathbf{X}_0} + \boldsymbol{\nu}^T \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}_0} \tag{2.61}$$

$$\mathbf{\Lambda}_f = \frac{\partial \Phi}{\partial \mathbf{X}_f} - \boldsymbol{\nu}^T \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}_f} \tag{2.62}$$

$$-\frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \frac{\partial \mathcal{H}_k}{\partial t_0} + \frac{1}{2} \sum_{k=1}^{N} w_k \mathcal{H}_k = \frac{\partial \Phi}{\partial t_0} - \boldsymbol{\nu}^T \frac{\partial \boldsymbol{\phi}}{\partial t_0} \tag{2.63}$$

$$\frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \frac{\partial \mathcal{H}_k}{\partial t_f} + \frac{1}{2} \sum_{k=1}^{N} w_k \mathcal{H}_k = -\frac{\partial \Phi}{\partial t_f} + \boldsymbol{\nu}^T \frac{\partial \boldsymbol{\phi}}{\partial t_f} \tag{2.64}$$

$$\mu_{jk} = 0, \text{ when } \quad g_{jk} < 0 \tag{2.65}$$

$$\mu_{jk} \leq 0, \text{ when } \quad g_{jk} = 0 \tag{2.66}$$

for $k = 1, \ldots, N$ and $j = 1, \ldots, c$. The final two equations that are required (in order to link the initial and final state and costate, respectively) are

$$\mathbf{X}_f = \mathbf{X}_0 + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \mathbf{f}_k \tag{2.67}$$

$$\mathbf{\Lambda}_f = \mathbf{\Lambda}_0 + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \left(-\frac{\partial L_k}{\partial \mathbf{X}_k} - \mathbf{\Lambda}_k^T \frac{\partial \mathbf{f}_k}{\partial \mathbf{X}_k} + \boldsymbol{\mu}_k^T \frac{\partial \mathbf{g}_k}{\partial \mathbf{X}_k} \right) \tag{2.68}$$

The total number of equations in set of discrete necessary conditions of Eqs.(2.57)-(2.68) is $(2n + m + c)N + 4n + q + 2$ (the same number of unknown variables). Solving these nonlinear algebraic equations would be an indirect solution to the optimal control problem.

2.2.4.4 Costate Estimate

One of the key features of the Gauss pseudospectral method is the ability to map the KKT multipliers of the NLP to the costates of the continuous-time optimal control problem. In particular, using the results of Sec.(2.2.4.2) and Sec.(2.2.4.3), a costate estimate for the continuous Bolza problem can be obtained at the Legendre-Gauss points

and the boundary points. This costate estimate is summarized below via the *Gauss Pseudospectral Costate Mapping Theorem:* (Ref.[7]).

Gauss Pseudospectral Costate Mapping Theorem

The Karush-Kuhn-Tucker (KKT) conditions of the NLP are exactly equivalent to the discretized form of the continuous first-order necessary conditions of the continuous Bolza problem when using the Gauss pseudospectral discretization. Furthermore, a costate estimate at the initial time, final time, and the Legendre-Gauss points can be found from the KKT multipliers, $\tilde{\mathbf{\Lambda}}_k$, $\tilde{\boldsymbol{\mu}}_k$, $\tilde{\mathbf{\Lambda}}_F$, and $\tilde{\boldsymbol{\nu}}$,

$$\begin{gathered}\mathbf{\Lambda}_k = \frac{\tilde{\mathbf{\Lambda}}_k}{w_k} + \tilde{\mathbf{\Lambda}}_F \quad , \quad \boldsymbol{\mu}_k = \frac{2}{t_f - t_0}\frac{\tilde{\boldsymbol{\mu}}_k}{w_k} \quad , \quad \boldsymbol{\nu} = \tilde{\boldsymbol{\nu}} \\ \mathbf{\Lambda}(t_0) = \tilde{\mathbf{\Lambda}}_0 \quad , \quad \mathbf{\Lambda}(t_f) = \tilde{\mathbf{\Lambda}}_F\end{gathered} \tag{2.69}$$

Using the substitution of Eq.(2.69), it is seen that Eqs.(2.40)-(2.52) are exactly the same as Eqs.(2.57)-(2.68).

The previous theorem indicates that solving the NLP derived from the Gauss pseudospectral transcription of the optimal control problem is equivalent to applying the Gauss pseudospectral discretization to the continuous-time variational conditions. Fig. 2.2 shows the solution path for both the direct and indirect methods.

2.2.4.5 Computation of Boundary Controls

It is seen in the GPM that the control is discretized only at the LG points and is *not* disctretized at either the initial or the terminal point. Consequently, the solution of the NLP defined by Eqs.(2.34), (2.35), (2.36), (2.37), and (2.38) does not produce values of the controls at the boundaries. The ability to obtain accurate initial and terminal controls can be important in many applications, particularly in guidance where real-time computation of the initial control is of vital interest.

First, recalling the augmented Hamiltonian, $\tilde{\mathcal{H}}_a$, for the continuous-time optimal control problem, we have

$$\tilde{\mathcal{H}}_a(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \equiv g + \boldsymbol{\lambda}^T \mathbf{f} - \boldsymbol{\mu}^T \mathbf{g} \tag{2.70}$$

where shorthand notation is used. Now recall that, from the minimum principle of Pontryagin, at every instant of time the optimal control is the control $\mathbf{u}^*(\tau) \in \mathcal{U}$ that

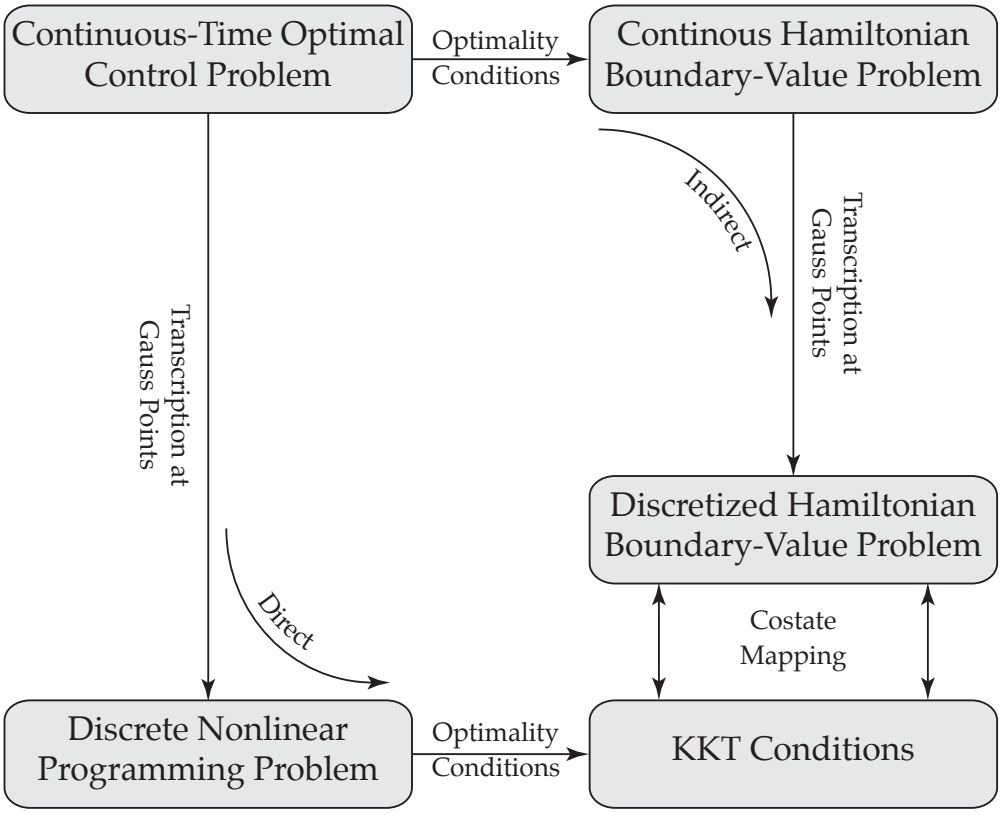


FIGURE 2.2: Equivalence of indirect and direct forms using the Gauss pseudospectral discretization.

satisfies the condition

$$\tilde{\mathcal{H}}_a(\mathbf{x}^*, \mathbf{u}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \leq \tilde{\mathcal{H}}_a(\mathbf{x}^*, \mathbf{u}, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \tag{2.71}$$

where $\mathcal{U}$ is the feasible control set. Consequently, for a given instant of time τ where $\mathbf{x}^*(\tau)$, $\boldsymbol{\lambda}^*(\tau)$, and $\boldsymbol{\mu}^*(\tau)$ are known, Eq.(2.71) is a constrained optimization problem in the $\mathbf{u}(\tau) \in \mathbb{R}^m$. In order to solve this constrained optimization problem at the *initial* time, it is necessary to know $\mathbf{x}^*(\tau_0)$, $\boldsymbol{\lambda}^*(\tau_0)$, and $\boldsymbol{\mu}^*(\tau_0)$.

Consider now the information that can be obtained by solving the NLP associated with the GPM. In particular, the primal solution to the NLP produces $\mathbf{X}(\tau_0)$ while the dual solution to the NLP can be manipulated algebraically to obtain the initial costate, $\boldsymbol{\Lambda}^*(\tau_0)$. However, because the NLP does not evaluate that path constraint at the boundaries, there is no associated Lagrange multiplier $\tilde{\boldsymbol{\mu}}(\tau_0)$. This apparent impediment can be overcome by applying the minimum principle in a manner somewhat different from that given in Eq.(2.71). In particular, suppose we let $\mathcal{H}$ be the *Hamiltonian* (not the augmented Hamiltonian), where $\mathcal{H}$ is defined as

$$\mathcal{H}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) \equiv L + \boldsymbol{\lambda}^T \mathbf{f} \tag{2.72}$$

It is seen in Eq.(2.72) that the term involving the path constraint is not included. The path constraint is instead incorporated into the feasible control set. In particular, suppose we let $\mathcal{V}_0$

$$\mathcal{V}_0 = \mathcal{U} \bigcap \mathcal{C}_0 \tag{2.73}$$

where $\mathcal{V}_0$ is the intersection of the original set of feasible controls at time τ_0, denoted $\mathcal{U}$, with the set of all controls at time τ_0 that satisfy the inequality constraint of Eq.(2.38), denoted $\mathcal{C}_0$. Then, using the values of $\mathbf{X}(\tau_0)$ and $\mathbf{\Lambda}(\tau_0)$, the following modified optimization problem in m variables $\mathbf{U}(\tau_0) \in \mathbb{R}^m$ can be solved to obtain the initial control, $\mathbf{U}(\tau_0)$:

$$\begin{aligned} \text{minimize} \quad & \mathcal{H}(\mathbf{X}(t_0), \mathbf{U}(\tau_0), \mathbf{\Lambda}(\tau_0), \tau_0; t_0, t_f) \\ \mathbf{U}(\tau_0) \in \mathcal{V}_0 \end{aligned} \tag{2.74}$$

It is noted that, because $\mathcal{V}_0$ is restricted by the inequality path constraint at τ_0, the solution of $\mathbf{U}(\tau_0)$ is equivalent to the solution of the following problem:

$$\begin{aligned} \text{minimize} \quad & \mathcal{H}(\mathbf{X}(\tau_0), \mathbf{U}(\tau_0), \mathbf{\Lambda}(\tau_0), \tau_0; t_0, t_f) \\ \mathbf{U}(\tau_0) \in \mathcal{U} & \\ \text{subject to} & \\ & \mathbf{g}(\mathbf{X}(\tau_0), \mathbf{U}(\tau_0), \tau_0; t_0, t_f) \leq \mathbf{0} \end{aligned} \tag{2.75}$$

Interestingly, if the constraint is *active*, then the initial path constraint multiplier, $\tilde{\boldsymbol{\mu}}(\tau_0)$, will also be determined by the minimization problem of Eq.(2.75). Finally, as alluded to above, the control at the terminal time, $\mathbf{U}(\tau_f)$, can be obtained by solving the minimization problem of Eq.(2.75) at $\tau = \tau_f$, i.e.

$$\begin{aligned} \text{minimize} \quad & \mathcal{H}(\mathbf{X}(\tau_f), \mathbf{U}(\tau_f), \mathbf{\Lambda}(\tau_f), \tau_f; t_0, t_f) \\ \mathbf{U}(t_f) \in \mathcal{U} & \\ \text{subject to} & \\ & \mathbf{g}(\mathbf{X}(\tau_f), \mathbf{U}(\tau_f), \tau_f; t_0, t_f) \leq \mathbf{0} \end{aligned} \tag{2.76}$$

2.2.5 Choosing a method

Choosing a method for solving an optimal control problem is based largely on the type of problem to be solved and the amount of time that can be invested in coding. An indirect approach has the advantage that it is simple to understand and produces highly accurate solutions when it converges. However, it is extremely sensitive to the unknown boundary conditions and it requires the derivation of the first-order optimality conditions of the optimal control problem. While for a simple problems it may be possible to derive the first order optimality conditions, in other problems it could be tedious, error-prone,

or even impossible (i.e.problems with table lookups). Hence, implementing the code is more difficult.

The accuracy and robustness of a direct method is highly dependent upon the direct method used. Direct shooting methods are really good for problems where the control can be parametrized in some manner and the problem can be well described using a small set of parameters.

As the difficulty increases a more accurate method is required. The two main reason that direct collocation methods work so well is because highly complex problems can be formulated and solved with today's NLPs. The reason that the NLP solvers can handle with these complexity is because they are designed to converge with poor initial guesses (e.g. constant controls) and are extremely computationally efficient because they exploit sparsity of the derivatives in the constraints and objective function.

In the Fig.(2.3) the main advantages and disadvantages of each method are summarized .

INDIRECT METHODS		DIRECT METHODS	
PROS	**CONS**	**PROS**	**CONS**
* High Accuracy * Solution satisfies necessary optimality conditions	*Necessary optimality conditions must be derived analitically *Small Radius of convergence : Need a good initial guess * Need a guess for costate	* Larger Radius of convergence. * No need to compute costate information * No need to derive analitically *Use high efficient NLP solvers	* Not as accurate as Indirect methods. * Do not produce costate information (except from peseudospectral)

FIGURE 2.3: Advantages and disadvantages of the methods to solve OCPs

Pseudospectral methods, allow to obtain accurate solution with relatively few discretization points. They are known to converge exponentially for problems whose solutions are smooth. Moreover, Gauss Pseudospectral method has established costate estimation procedures which can be used to verify the optimality of the resulting solution.

2.3 Numerical tools used in Optimal Control Problems

At the heart of a well-founded method for solving optimal control problems are the following three fundamental components: (1)methods for solving the differential equations and integrating functions; (2) a method for solving a system of nonlinear algebraic equations; and (3) a method for solving a nonlinear optimization problem. Methods for solving solving differential equations and integrating functions are required for all numerical methods in optimal control. In an indirect method, the numerical solution of differential equations is combined with the numerical solution of systems of nonlinear equations, while in a direct method, the solution of differential equations is combined with nonlinear optimization.

2.3.1 Nonlinear Programming

Essentially all numerical methods for solving the Optimal Control Problems incorporate some type of iterations with a finite set of unknowns. In fact, progress in optimal control solution methods closely parallels the progress made in the underlying Non-Linear Programming methods (NLP). In this section theory under this methods will be explained, as well as the most widely used *gradient based methods* will be presented.

The NLP may either be dense (i.e. a large percentage of the derivatives of the objective function and the constraint functions with respect to $\boldsymbol{z}$ are non-zero) or maybe sparse (i.e. a large percentage of the derivatives of the objective function and the constraint functions with respect to $\boldsymbol{z}$ are zero). Dense NLP's are typically small while sparse NLP are extremely large. Sparsity is a key feature that is extremely important to exploit, because they can affect both the convergence and computational cost. In particular, the vast majority of the derivatives of the defect and path constraints in a direct collocation method (Sec.2.3.2.2) are *zero*. Because the constraint Jacobians are very large (thousands up to ten thousands variables), it is absolutely essential that only the nonzero derivatives be computed and stored. The different NLP solvers exploit this characteristics.

Numerical methods for solving NLP's fall into two main categories: *gradient based methods* and *heuristic methods*. We will focus on gradient based methods.

In a iterative *gradient based method*, an *initial guess* is made of the unknown decision vector $\boldsymbol{z}$. At every k-iteration, a search direction, $\boldsymbol{p}_k$ and a step length α_k is determined. The search direction provides a direction in $\mathbb{R}^{n_z}$ along which to change the current value $\boldsymbol{x}_k$ while the step length provides the magnitude of that change. The update from $\boldsymbol{z}_k$

to $\boldsymbol{z}_{k+1}$ is performed as follows:

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \alpha_k \boldsymbol{p}_k \tag{2.77}$$

In the case of minimization, the search direction is chosen to "sufficiently decrease" the objective function in the form

$$J(\boldsymbol{z}_{k+1}) \leq J(\boldsymbol{z}_k) + K\alpha_k \nabla J^T(\boldsymbol{z}_{k+1})\boldsymbol{p}_k \tag{2.78}$$

and K is a parameter between 0 and 1.

The use the following four basic components (Ref.[10]):

1. **Approximate Subproblem**: The is an approximation of Eq.(2.12) around the current iterate z_k that can be solved (approximately), and produces a step or search direction that improves the current estimate z_k. Examples of such subproblems are linear or quadratic programming approximation, the barrier problem and an augmented Lagrangian.

2. **Global Convergence Strategy**: The (local) subproblem alone cannot guarantee convergence to a stationary point, and optimization methods need a convergence strategy that forces the sequence z_k to converge. Examples of convergence strategies are penalty functions, augmented Lagrangian, filter, and funnel methods.

3. **Global Convergence Mechanism**: The global convergence mechanism that improve the subproblem by shortening the step that is computed. There are two classes of convergence mechanism: line search and trust region.

4. **Convergence Test**: A convergence test is needed to stop the algorithm once the error in the KKT conditions is below a user-specified tolerance. In addition, solvers may converge only to a local minimum of the constraint violation.

The most commonly used *gradient based method* are *Sequential Quadratic Programming* (SQP) and *Interior Point* methods (IP).

- ***Sequential Quadratic Programming*** (SQP): is an iterative method for nonlinear optimization based on solving several easier problems. SQP methods are used on problems for which the objective function and the constraints are twice continuously differentiable. This method is equivalent to applying Newton's method to the first-order KKT optimality conditions. Hence the search direction $\boldsymbol{p}$ is determined by solving s *Quadratic Program* subproblem, where it is to determine $\boldsymbol{p}$ in order to minimize the following function

$$\boxed{\begin{array}{ll} \textit{Minimize} & \frac{1}{2}\boldsymbol{p}^T H \boldsymbol{p} + \nabla J(\boldsymbol{z})\boldsymbol{p} \\ \\ \textit{Subject to} & \nabla c(\boldsymbol{z})\boldsymbol{p} + c(\boldsymbol{z}) = 0 \\ & \nabla h(\boldsymbol{z})\boldsymbol{p} + h(\boldsymbol{z}) \leq 0 \end{array}} \tag{2.79}$$

where H is the Hessian of the Lagrangian (Eq.(2.19)). It can be obtained as

$$H = \nabla^2_{zz}\mathcal{L} \tag{2.80}$$

Examples of well-known software that uses SQP methods include the dense NLP solver **NPSOL** and the sparse NLP solver **SNOPT** (Ref.[19]). These NLP solvers are available in a variety of implementations, including Fortran, C++, Matlab®.

- ***Interior point method*** (IP): An iterative method also known as barrier method for solving optimization problems based on the idea of combining the objective function and constraints into a merit function to attack the problem by solving a sequence of unconstrained problems (Ref.[10]). The barrier maintains the variables in the strict interior of the inequalities preventing them from reaching the boundary because it increases its value as the variables get closer to the boundaries. As an example, a *barrier function* usually takes the form of a *log barrier form*

$$\boxed{\textit{Minimize} \qquad B(\boldsymbol{x}) = J(\boldsymbol{x}) - \mu \sum_{i=1}^{m} \log(c_i(\boldsymbol{x}))} \tag{2.81}$$

where μ is a positive parameter and control the magnitude of the barrier term. The barrier function is then minimized for a sequence of optimization problems. The solution to these problems will approach the solution to the original problem as μ approaches zero. The problem arises when the optimal solution lies on the barrier, because we could not find it.

Well Known sparse IP methods include **LOQO** and **IPOPT** (Ref.[20]). These NLP solvers are available in a variety of implementations, including Fortran, C++, Matlab®.

In this project, the solvers SNOPT and IPOPT will be used.

2.3.1.1 SNOPT

Algorithm Methodology: SNOPT (Gill et al.,2006 [19]) implements an SQP algorithm, suitable for large, spare problems. The Hessian of the Lagrangian is updated by using limited-memory quasi-Newton updates. SNOPT solves (minor

iterations) each QP using SQOPT, which is a reduced Hessian active-set method. It includes an option for using a projected conjugate gradient method rather than factoring the reduced Hessian. SNOPT starts by solving an "elastic program" that minimizes the constraint violation of the linear constraints of Eq.(2.12). The solution to his program is used as a starting point for the major iterations. If a QP subproblem is found to be infeasible or unbounded, the SNOPT tries to solve an elastic problem that correspond to a smooth reformulation of the l_1-exact penalty function. The solution from major iterations is used to obtain a search direction along which an augmented Lagrangian merit function is minimized.

Software and Technical Details: SNOPT is implemented in Fortran77 ans is compatible with newer Fortran compilers. All function in the SNOPT library can be used in parallel or by multiple threads. It can be called from other programs written in C and Fortran, such as AMPL, GAMS, AIMMS and MATLAB. The user or the interface had to provide routines that evaluates functions values and gradients. When gradients are not available, SNOPT uses finite differences to estimate them. SNOPT is available under commercial and academic licenses (300 variables limit) from Standford Business. Software Inc.

2.3.1.2 IPOPT

Algorithm Methodology: IPOPT is line-search filter interior-point method (Ref.[20]). The outer loop approximately minimizes a sequence of non-linearly equality constrained barrier problems for a decreasing sequence of barrier parameters. The inner loop uses line-search filter SQP method to approximately solved each barrier problem. Global convergence os a barrier problem is enforced through a line-search filter method, and the filter is reset after each barrier parameter update. The inner iterations includes second-order correction steps and mechanism for switching to a feasibility restoration if the step size becomes to small. The solver has an option for using limited memory BFGS updates to approximate the Hessian of the Lagrangian.

Software and Technical Details: The solver is written in C++ and has interfaces to C, C++, Fortran, AMPL, MATLAB. The user (or the modelling language) must provide function and gradient information. and possibly the Hessian of the Lagrangian (in sparse format). IPOPT is available in COIN-OR under Common Public License at its website.

2.3.2 Integration of functions

Integrating the differential equations for the states in direct methods and for both the states and costates in indirect methods is needed to find the solution of the OCP. Moreover, when the cost is given in the Bolza form, it can be converted to the Majer form by adding an additional state x_{n_x+1} and adding the differential equation:

$$\dot{x}_{n_x+1} = L[\boldsymbol{x}(t), \boldsymbol{u}(t), t;] \tag{2.82}$$

with the initial conditions $x_{n_x+1}(t_0) = 0$. Then, the cost function (Eq.(2.1)) would be given in Majer form as

$$\mathcal{J} = \Phi[\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f; \mathfrak{p}] + x_{n_x+1}(t_f) \tag{2.83}$$

A common scheme would then be used to solve the complete system of differential equations.

Consider the Initial-Value Problem (IVP) (Ref.[17]) :

$$\dot{x} = f(\boldsymbol{x}(t), t), \qquad x(t_0) = x_0 \tag{2.84}$$

Furthermore, consider a time interval $[t_i, t_{i+1}]$ over which the solution to the differential equation is desired. In other words, given the value of the state at t_i, $\boldsymbol{x}(t_i) = \boldsymbol{x}_i$ it is desired to obtain the state at t_{i+1}, $x(t_{i+1}) = x_{i+1}$. Integrating Eq.(2.84), we can write :

$$x_{i+1} = x_i + \int_{t_i}^{t_{i+1}} \dot{x}(s)ds = x_i + \int_{t_i}^{t_{i+1}} f(x(s), s)ds \tag{2.85}$$

We will consider now two approaches for solving differential equations: time marching and collocation.

2.3.2.1 Time-Marching

In a time marching method, the solution of the differential equation at each time step t_i is obtained sequentially using current and/or previous information about the solution. Time-marching methods are divided into two categories: (1) multiple-step methods and (2) multiple-stage methods.

- **Multiple- Step Methods** : In a *multiple-step method*, the solution at each time step t_i is obtained from a defined set of values $t_{i-j}, ..., t_i$ where j is the number of

steps. The general form can be written as follows

$$x_{i+j} = \sum_{k=0}^{j} \alpha_k x_{i+k} + h \sum_{k=0}^{j} \beta_k \boldsymbol{f}_{i+k} \tag{2.86}$$

where α_k and β_k are known constants, and h is the step size. If $\beta_k = 0$, the method is explicit; otherwise it is implicit. When employing a implicit method, the solution at t_{i+1} is obtained using information at $(j-1)$ previous points. Clearly, this implies some method must be used to start the process such as a predictor-corrector method, where the corrector is normally an implicit method. Implicit methods are more stable than explicit methods, but and implicit method requires more computation at each step due to solve the implicit equation of the state. The two most commonly used multiple steps methods are the *Adams-Bashforth* and *Adams-Moulton*. All well-implemented schemes have some mechanism for adjusting the integration step size to control the integration error.

- **Multiple-stage Methods** : It is a class of one-step method where we divide the interval $[t_i, t_{i+1}]$ into K subintervals $[\tau_j, \tau_{j+1}]$ where

$$\tau_j = t_i + h_i \alpha_j, \qquad (j = 1, ..., K), \qquad h_i = t_{i+1} - t_i \tag{2.87}$$

and $0 \leq \alpha_j \leq 1, (j = 1, ..., K)$. Each value τ_j is called a *stage*. The integral from t_i to $t_i + 1$ can be approximated via a quadrature as

$$\int_{t_i}^{t_{i+1}} f(x(s), s) ds \approx h_i \sum_{j=1}^{K} \beta_j f(x_j, \tau_j) \tag{2.88}$$

In the previous equation it is seen that the values of the state at each stage are required in order to evaluate the quadrature approximation. These intermediate values are obtained as

$$x(\tau - j) - x(t_i) = \int_{t_i}^{\tau_j} f(x(s), s) ds \approx h_i \sum_{l=1}^{K} \gamma_{jl} f(x_l, \tau_l) \tag{2.89}$$

If we combine the last two equations, we obtain the family of K stage *Runge-Kutta* methods.

$$\int_{t_i}^{t_{i+1}} f(x(s), s) ds \approx h_i \sum_{j=1}^{K} \beta_j f_{ij}, \qquad f_{ij} = f(x_i + h_i \sum_{l=1}^{K} \gamma_{jl} f_{il}, t_i + h_i \alpha_j) \tag{2.90}$$

where α_j, β_j and γ_j are known constants, and h_i is the step size.

Finally, we have to consider that the selected method for the integration has to be consistent, stable and convergent for the differential system considered.

2.3.2.2 Collocation

Collocation methods enforce the dynamic equations through quadrature rules or interpolation. Suppose over a subinterval $[t_i, t_{i+1}]$, we choose to approximate the state using the following K^{th}-degree piecewise polynomial

$$\boldsymbol{x}(t) \approx \boldsymbol{X}(t) = \sum_{k=0}^{K} a_i (t - t_i)^k, \qquad t \in [t_i, t_{i+1}] \tag{2.91}$$

These coefficients $(a_0, ..., a_K)$ of the interpolating function (interpolant) are posed such that it passes through the state values and maintains the state derivatives at the nodes spanning one interval (or subinterval) of time. The interpolant is then evaluated at points between the nodes, called collocation points. The collocation points internal to a subinterval used to formulate an integration rule are chosen to increase the order of accuracy to the highest order possible. At each collocation point τ_j, an equality constraint or defect contraint ζ_j is formed, equating the interpolant derivative to the state derivative function, thus ensuring that the equations of motion hold (approximately) true across the entire interval of time

$$\mathcal{C}_j = \dot{\boldsymbol{X}}(\tau_j) - \boldsymbol{f}(x(\tau_j), \tau_j) \tag{2.92}$$

The defect constraints can then be stacked into a matrix as

$$\mathcal{C} = [\mathcal{C}_1, \ldots, \mathcal{C}_M] \tag{2.93}$$

Then , the objective of these methods is to determine the coefficients that set the *defect constraints* equal to zero ($\mathcal{C} = 0$) using a proper root finding technique. This equation is called a *collocation equation*.

One of the simplest methods of collocation is the Hermite-Simpson collocation method. This method is so called because a third-order Hermite interpolating polynomial is used locally within the entire sequence of intervals, each solved at the endpoints of an interval and collocated at the midpoint. When arranged appropriately, the expression for the collocation constraint corresponds to the Simpson integration rule. A generalization of the method to use the n-th order Hermite interpolating polynomial, and choosing to take the nodes and collocation points from a set of Legendre-Gauss-Lobatto points

defined within the local time intervals, gives rise to the HLGL collocation method. Other collocation methods are based, for instance, on Gauss or Radau collocation schemes.

According to this, collocation methods fall into three different categories: *Gauss Methods*, *Radau Methods*, and *Lobatto Methods*. In a *Gauss Method*, neither of the end-points t_i or t_{i+1} are collocation points. In a *Radau Method*, at most one of the end-points, t_i or t_{i+1} is a collocation point. In a *Lobatto Method* both of the endpoints t_i and t_{i+1} are collocation points.

Finally, one of the key difference between collocation and time-marching is that in collocation is not necessary a predictor-corrector because the value of the state at each discretization point is being solved simultaneously.

Third-degree Gauss-Lobatto integration rule

The Hermite-cubic polynomial representing the time history between the endpoint t_i and t_{i+1} may be constructed such both the values and first derivatives of the interpolant polynomial coincide with the values and first derivatives of function $f(\boldsymbol{x})$ at the extremes of the interval.

$$0 = t_1 < t_2 < \cdots < t_N = t_f \tag{2.94}$$

$$\boldsymbol{x}(t) \approx \boldsymbol{X}(t) = \sum_{K=0}^{3} c_K^i (\frac{t - t_i}{h_i})^K, \qquad t_i < t < t_{i+1}, \quad i = 0, \ldots, N-1 \tag{2.95}$$

$$c_0^i = \boldsymbol{x}(t_i) \tag{2.96}$$

$$c_1^i = h_i f_i \tag{2.97}$$

$$c_2^i = -3\boldsymbol{x}(t_i) - 2h_i f_i + 3\boldsymbol{x}(t_{i+1}) - h_i f_{i+1} \tag{2.98}$$

$$c_3^i = 2\boldsymbol{x}(t_i) + h_i f_i - 2\boldsymbol{x}(t_{i+1}) + h_i f_{i+1} \tag{2.99}$$

$$where f_i = f(\boldsymbol{x}(t_i), t_i), h_i = t_{i+1} - t_i \tag{2.100}$$

In this case, the center $t_{c,i} = t_{i+1/2} = (t_i + t_{i+1})/2, i = 1, \ldots, N-1$ of the each subinterval is set as a collocation point. Thus, the numerical approximation of the state $X_{c,i}$ is

$$X_{c,i} = \frac{1}{2}(\boldsymbol{X}_i + \boldsymbol{X}_{i+1}) + \frac{h_i}{8}(f(\boldsymbol{X}_i) - f(\boldsymbol{X}_{i+1})); \tag{2.101}$$

This equations have to satisfy the differential equations at the collocations points. The corresponding defect constraints can be expressed as follows :

$$\mathcal{C}_i = \boldsymbol{X}_i - \boldsymbol{X}_{i+1} + \frac{h_i}{6}(f(\boldsymbol{X}_i) + 4f(\boldsymbol{X}_{c,i}) + f(\boldsymbol{X}_{i+1})) \qquad for \qquad i = 1, \ldots, N-1 \tag{2.102}$$

2.3.3 System of nonlinear algebraic equations

Solving a system of nonlinear algebraic equations is equivalent to *root-finding*. In the case where all of the algebraic equations can be written as equalities, we have a problem of the form:

$$g(\boldsymbol{z}) = 0 \tag{2.103}$$

The most common method for multidimensional root-finding is *Newton's method*. In Newton's method, an initial guess is made of the vector $\boldsymbol{z}$. The iterates are then computed as

$$\boldsymbol{z}_{k-1} = \boldsymbol{z}_k + (\frac{\partial g}{\partial \boldsymbol{z}})^{-1}_{z_k} g(\boldsymbol{z}_k) \tag{2.104}$$

As it can be inferred from Eq.(2.104), the method breaks down when the Jacobian of g ($i.e. \partial g/\partial \boldsymbol{z}$) is singular. In order to ensure that the iterations do not become undefined, the gradient is normally approximated via a quasi-Newton approximation.

In fact, one way of using a quasi-Newton method to find the root of a system, is via an NLP solver. In particular, Eq. (2.103) can be solved using an NLP solver such as *SNOPT*(Ref.[19]) or *IPOPT* (Ref.[20]), by using a fictitious cost function such us:

$$\mathcal{J}(\boldsymbol{z}) = (\boldsymbol{z} \quad \boldsymbol{z}_0)^T (\boldsymbol{z} \quad \boldsymbol{z}_0) \tag{2.105}$$

where z_0 is the initial guess. The optimization problem is then minimize Eq.(2.105) subject to the constraints of Eq.(2.103). The iterations will be then will be then proceed via the corresponding update procedure, as explained in Sec.(2.3.1).

2.3.4 Calculating Gradients

From the KKT conditions in 2.13, we can easily infer that the computation of gradients is a crucial task in the determination of the Optimal solution by the NLP solver. Perhaps, the most obvious way to compute this derivatives is by analytical differentiation. While such an approach is appealing because analytical derivatives are exact and generally result in a faster optimization, for much software development it is impractical to compute derivatives analytically. As a result, alternative means must be employed to obtain the necessary gradients. The numerical methods fall into three categories: *Finite Differences*, *Automatic Differentiation* and *Complex step differentiation*.

- **Finite differences** (FD) : The two more common methods for the finite-difference approximation of a derivative are *forward differencing* and *central differencing*. A forward and central difference approximation to the derivative of a function $f(x)$

are given respectively :

$$\frac{df}{dx} \approx \frac{f(x+\delta) - f(x)}{\delta} \tag{2.106}$$

$$\frac{df}{dx} \approx \frac{f(x+\delta) - f(x-\delta)}{2\delta} \tag{2.107}$$

where δ is perturbation that is sufficiently small to provide a good approximation to the derivative. While finite differences approximations work for solving some NLP's, for many others using these approximation will result in either slow convergence to a solution or non-convergence. Because finite differences has limits, more accurate methods have been developed. The accuracy for forward differencing turns out to be $\mathcal{O}(\delta)$, whereas for central differentiation $\mathcal{O}(\delta^2)$ and extremely small values of δ cannot be used, because of cancellation errors.

- **Complex-step Differentiation** : As it names implies, it requires that the programming language allow for complex arithmetic(as Matlab®). The basic idea behind complex -step is as follows. Consider an analytic complex function $f = u + iv$ of the complex variable $x + iy$. It is known that f satisfies the *Cauchy-Riemann* equations

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \qquad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \tag{2.108}$$

Then

$$\frac{\partial u}{\partial x} = \lim_{\delta \longrightarrow 0} \frac{u(x + i(y+\delta)) - v(x+iy)}{\delta} \tag{2.109}$$

where δ is real. Now, considering, $y = 0$, $u(x) = f(x)$ and $v(x) = 0$ we obtain

$$\frac{df}{dx} \approx \frac{Im(f(x+i\delta))}{\delta} \tag{2.110}$$

The accuracy turns out to be is $\mathcal{O}(\delta^2)$, and extremely small values of δ can be used, because it does not suffer from cancellation errors.

- **Automatic Differentiation** (AD): Automatic Differentiation is the generic name for techniques that use the computational representation of a function to produce analytic values for the derivatives. In automatic differentiation, the derivatives of the functions are computed to the same precision as they would be if they had been computed analytically. Two commonly used approaches to AD are *source transformation* and *function overloading*. In a*source transformation* approach, the code used to compute the functions is transformed into a new code that is used to compute the derivatives. It has been well developed in the program **ADIFOR** and the Matlab package *myad*. An alternative to *source transformation* that takes advantage of modern computer programming languages such as C++ and Matlab® is *function overloading*. In this method, a class is defined that operate the objects

of the class. Because the language permits function overloading, the build-in functions can be defined to output both the function value and it derivatives. It has been successfully implemented in the Matlab package *adiMat*

They both are founded on the observation that any function, no matter how complicated, is evaluated by performing a sequence of simple elementary operations (e.g. addition, multiplication, power, trigonometric or logarithmic functions).In this manner, the chain rule for differentiation can be implemented in the same number of operations as would be required to compute the derivatives by hand. There are two basic modes of automatic differentiation : the *forward* and *reverse* modes. For further information in AD see Ref.[10].

Errors in gradient vectors could lead to three difficulties, greatly affecting the performance of any gradient-based optimization method. First, inaccurate gradients lead to the calculation of search directions that may not be descent directions for the nonlinear program, leading to premature termination of the algorithm. Second, inaccurate derivative information can lead to Hessian (or approximate Hessians) that are ill-conditioned, also leading to poor search directions. Finally, and most importantly, even if the algorithm finds the solution, accurate derivatives are needed to certify that it is the optimum.

2.3.5 Calculating Hessians

Hessians can be mainly computed by hand, using a quasi-Newton approximation or by Automatic differentiation. Before AD, the most common and efficient method was the quasi-Newton approximation ($\mathbf{W}_k$) in the form of an update method, called the BFGS (Broyden-Fletcher-Goldfarb-Sahnno) method (Ref.[10]). The BFGS for the case of unconstrained problem is given as follows. At the first step in the optimization,$\mathbf{W}_k$ is generally set to the identity matrix. Then, at each step in the optimization, $\mathbf{W}_k$ is updated as

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \frac{v_k v_k^T}{v_k^T s_k} - \frac{\mathbf{W}_k s_k s_k^T \mathbf{W}_k}{s_k^T \mathbf{W}_k s_k} \tag{2.111}$$

where $v_k = \nabla f(\boldsymbol{z}_{k+1}) - \nabla f(\boldsymbol{z}_k), \boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \boldsymbol{p}_k s_k$, and s_k is obtained by solving the equation:

$$\mathbf{W}_k s_k = -\nabla f(\boldsymbol{z}_k) \tag{2.112}$$

Although the development of AD has diminished the appeal of quasi-Newton methods, they still remain competitive in many types of problems.

In the case of solving large problems whose Hessians matrices cannot be computed at a reasonable cost or are too dense to be manipulated easily, Limited-memory quasi-Newton methods are quite appropriate. The most common is the L-BFGS updating

formula. These methods maintain simple and compact approximations of Hessian matrices: Instead of storing fully dense $n \times n$ approximations, they save just a few vectors of length n that represent the approximation implicitly. Despite this modest storage requirements, they often yield and acceptable rate of convergence (Ref.[10]).

2.3.6 Scaling

The objective and constraint function might be well scaled. From the standpoint of Newton-based solvers, good scaling is required for accurate solution of linear systems for determination of Newton steps. A widely used rule is to scale the objective, constraint functions, and the variables so that the magnitudes of the gradient elements are close to the unit.

Chapter 3

Special Perturbation Methods based on ideal frames

3.1 Introduction

In order to describe precisely or propagate over time the perturbed Keplerian motion of an orbiting object given the position and velocity at some initial time t_0, there exist three types of methods, depending on how the theory handles perturbing forces:

- **Special Perturbation theories** (SP) include a high-precision numerical integrator together with a relatively complete and accurate formulation of accelerations acting on a body. This theory is accurate but computationally resource consuming.
- **General Perturbation theories** (GP) are analytical solutions for perturbed orbital motions. The result is usually expressed in a closed-form solution in terms of algebraic and trigonometric functions of the orbital elements of the body. The elements at any prediction time can be found immediately, avoiding costly step-by-step numerical integration unlike in SP. In GP, two body equations are replaced with analytical approximations to certain degree and order. Consequently, they are less accurate and flexible than SP.
- **Semi-Analytical theory** is formulated by replacing the conventional equations of motion with : (1) equation of motion for the mean elements and (2) expression for the short periodic motion. This offers much larger integration steps. Because of its formulation the performance of this method lies between SP and GP in terms of accuracy and time consuming.

3.2 Special Perturbation Methods

This chapter will focus on high-precision numerical propagators, which are the most appropriate when accuracy is the premium, like in operations; and they are the best solution to model realistic problems. In this section, it will briefly presented the historical evolution of the numerical methods and both Deprit and Dromo formulations.

Special perturbation methods should be regular, robust and efficient. Regular means that the propagator is free of singularities. Robust means that it should be numerically stable. Efficient means that the propagator should render accurate results with low time-consumption and share a common formulation for elliptical, parabolic and hyperbolic problems.

In celestial mechanics there are bad-conditioned problems that are highly sensitive to initial conditions, where small errors, which could not have short-term relevance, turned into inadmissible divergence in the long term. The detailed dynamical simulation requires advanced numerical techniques, which should combine a proper treatment of the equations of motion with carefully elaborated numerical methods in order to minimize numerical errors. By this way, the quality of the solution would be considerably improved. This type of problems can be found in the traditional special perturbation methods (Cowell's and Enckes's method) where errors are propagated exponentially downstream. For long integration times, detailed simulation requires higher precision methods.

With the development of computers, the advantages of the method of variation of parameters (VOP), may outweigh, in many individual problems, the simplicity and generality of a straight forward numerical integration in Cartesian coordinates with respect to fixed frame (Cowell's method). These advantages include especially (Ref. [21]) (1) a lesser accumulation of errors because the integration processes parameters that are affected solely by the usually small perturbations, (2) fewer steps of integration because the parameters are slowly varying and (3) a deeper insight in the physics of the problem because the elements are selected to enhance the especial effects of the perturbing forces on the characteristics of a Keplerian motion.

There have been numerous attempts at finding the parameters best suited to the differential equations that describe the motion of a mass particle in a gravitational field. Classical astronomers have been overly concerned that conversion from Cartesian coordinates to Keplerian elements not raise the order of the six-order classical formulation. Actually, rising the order is not a big burden for computer algorithms and, in some cases, leads to substantial improvements of performance. Herrick was the first one to abandon this requirement and proposed a system of redundant orbital elements. Subsequently,

many authors contributed notably in this field, proposing new sets of elements, containing some of them vectorial elements instead of scalar magnitudes. Parameter should be carefully selected avoiding singularities (e.g. Euler angles at half turns, apsidal frame at small eccentricities).

Both presented methods (DROMO and DEPRIT) are based on the *ideal frame concept* proposed by Hansen (Ref.[22]). The Hansen's ideal frame is attached to the orbital frame and allows us to separate the periodic perturbation in the orbital plane from those of the orbital plane itself. The other important and related feature of the ideal frame is that the equations of the absolute motion have the same form, because the ficticious forces of Coriolis and centrifugal forces are equal and opposed.

In 1975 Deprit in Ref.[21] *proposed a new set of elements to describe Keplerian motion subject to perturbing forces. The resulting equations do not break down for small eccentricities, small inclinations and rotations of the ideal frame reference that are half turns. The parameters are selected wih a view of simplifying the programming of the right hand members.* This elements were called *ideal elements* and were based on the *ideal* frame concept of Hansen (Ref.[22]). The elements consists on eight parameters, whose independent variable is the physical time. The attitude of the ideal frame is described by quaternions, and consequently the system is redundant. Quaternions had been widely used in the analysis of the attitude dynamics of a rigid body, but they had not been commonly used in orbital mechanics until Ref.[21].

In year 2000 a house-made orbital propagator was developed by the SDG-UPM. The aim of the project was to develop a regular, robust and efficient propagator. To fullfill these requirements, a special perturbation method of variation of parameters based in a set of redundant variables including quaternions was considered. That propagator was called DROMO and initially it was mainly used in numerical simulations of electrodynamic tethers. The description of the movement is closely related to the *ideal frame* concept introduced by Hansen. The DROMO propagator consists on eight parameters whose independent variable is a fictitious time provided by a generalized Sundman transformation which reduces to the true anomaly when the motion is purely Keplerian. Seven of the eight variables are *generalized elements*, that is, they take constant values in the Keplerian motion.

DEPRIT and DROMO methods have many similarities (e.g. both use quaternions) and also many differences (e.g. Deprit is only applicable for elliptical orbits while DROMO do not depend on the orbit). In the next sections we will reach both formulations step by step and comparisons between them will be made.

3.3 Basic elements on ideal frame based methods

Let consider the two body problem, which is governed by the differential equation:

$$\frac{\mathrm{d}^2\hat{\boldsymbol{r}}}{\mathrm{d}\,t^2} = -\frac{\hat{\mu}}{\hat{r}^3}\,\hat{\boldsymbol{r}} + \hat{\boldsymbol{a}}_p$$

where $(\hat{\cdot})$ distinguishes dimensional variables and $\hat{\boldsymbol{a}}_p$ denotes the specific force or acceleration due to external perturbations (If the subscript p is not present, $\hat{a}$ means the semi-major axis of the ellipse) . Defining a certain characteristic distance, L_c, and characteristic time, t_c, the dimensionless variables:

$$\boldsymbol{r} = \frac{\hat{\boldsymbol{r}}}{L_c}, \qquad \tau = \frac{t}{t_c}, \qquad \boldsymbol{a}_p = \frac{\hat{\boldsymbol{a}}_p}{L_c/t_c^2}, \qquad \mu = \frac{\hat{\mu}}{L_c^3/t_c^2}$$

convert the governing equation of motion into:

$$\frac{\mathrm{d}^2\boldsymbol{r}}{\mathrm{d}\,\tau^2} = -\frac{\boldsymbol{r}}{r^3} + \boldsymbol{a}_p \tag{3.1}$$

To construct a proper normalization such that $\mu = 1$, hereafter $L_c = \hat{a}$ and $t_c = 1/\hat{n}$.

3.3.1 Orbital frame

Let $Oxyz$ be the orbital frame $(\mathcal{O})$ rotates attached to the particle M with the origin O in the main attracting body (Fig.(3.1)). The orbital frame is defined by the basis $\mathcal{O} : \lfloor \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k} \rceil$ such that: versor $\boldsymbol{i}$ is parallel to the radius-vector pointing zenithwards; versor $\boldsymbol{k}$ is described by the angular momentum vector, $\boldsymbol{h}$; and $\boldsymbol{j}$ completes an orthogonal dextral reference frame. That is:

$$\boldsymbol{i} = \frac{\boldsymbol{r}}{r}, \qquad \boldsymbol{k} = \frac{\boldsymbol{h}}{h}, \qquad \boldsymbol{j} = \boldsymbol{k} \times \boldsymbol{i} \tag{3.2}$$

The plane defined by $\boldsymbol{i}$ and $\boldsymbol{j}$ defined the *orbital frame*. In the non-perturbed Keplerian motion, it is fixed to the inertial frame $(\mathcal{I})$ and it contains the trajectory of the particle. In the perturbed motion the orbital plane is moving into the inertial frame but it contains the osculating orbit. Note that the orbital frame is not attached to the orbital plane. The motion of this frame relative to the orbital frame is a pure rotation about $\boldsymbol{k}$.

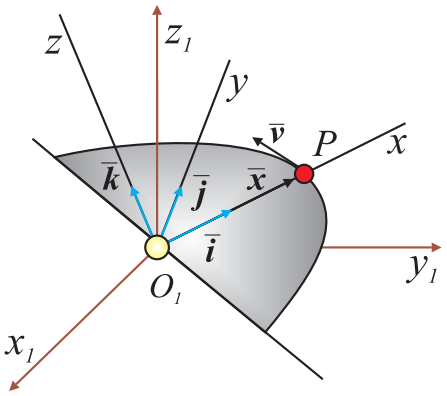


FIGURE 3.1: Orientation of the Orbital frame

3.3.2 Angular velocity of the orbital plane

Let h be the angular momentum defined by $\boldsymbol{h} = \boldsymbol{r} \times \boldsymbol{v}$, where $\boldsymbol{r}$ is the position vector, and $\boldsymbol{v}$ is the velocity vector. Differentiating this equation with respect to the non-dimensional time and considering Eq.(3.1) we obtain

$$\frac{\mathrm{d}\boldsymbol{h}}{\mathrm{d}\,\tau} = \boldsymbol{r} \times \boldsymbol{a}_p \tag{3.3}$$

Hence, in absence of perturbations ($a_p = 0$) the angular momentum remains constant.

Let consider the perturbing force and its components respect to the orbital frame

$$\boldsymbol{a}_p = a_{px}\boldsymbol{i} + a_{py}\boldsymbol{j} + a_{pz}\boldsymbol{k}$$

The time derivative of versors $(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k})$ respect to the inertial frame are :

$$\frac{\mathrm{d}\boldsymbol{i}}{\mathrm{d}\,\tau} = \frac{1}{r}[\boldsymbol{v} - \frac{\mathrm{d}r}{\mathrm{d}\,\tau}\boldsymbol{i}] = \frac{h}{r^2}\boldsymbol{j} = \boldsymbol{\omega}_{\mathcal{OI}} \times \boldsymbol{i} \tag{3.4}$$

$$\frac{\mathrm{d}\boldsymbol{k}}{\mathrm{d}\,\tau} = \frac{1}{h}[\frac{\mathrm{d}\boldsymbol{h}}{\mathrm{d}\,\tau} - \frac{\mathrm{d}h}{\mathrm{d}\,\tau}\boldsymbol{k}] = -\frac{r}{h}a_{py} = \boldsymbol{\omega}_{\mathcal{OI}} \times \boldsymbol{j} \tag{3.5}$$

$$\boldsymbol{j} = \boldsymbol{k} \times \boldsymbol{i} = \boldsymbol{\omega}_{\mathcal{OI}} \times \boldsymbol{k} \tag{3.6}$$

According to this, the motion of the orbital frame can be viewed as a rotation from the inertial frame with the angular velocity

$$\boldsymbol{\omega}_{\mathcal{OI}} = \frac{h}{r^2}\boldsymbol{k} + \frac{r}{h}a_p\boldsymbol{i} = \omega_{\mathcal{OH}} + \omega_{\mathcal{HI}} \tag{3.7}$$

The motion can be viewed as the product of two motions:

1) a rotation from the fixed frame $\mathcal{I}$ to an intermediate frame $\mathcal{H}$ defined by the basis $[\boldsymbol{i}_H, \boldsymbol{j}_H, \boldsymbol{k}_H]$ with the angular velocity

$$\boxed{\omega_{\mathcal{H}\mathcal{I}} = \frac{r}{h} a_{pz} \boldsymbol{i}} \tag{3.8}$$

2) a rotation from the intermediate frame $\mathcal{H}$ to the orbital frame $\mathcal{O}$ with the angular velocity

$$\boxed{\omega_{\mathcal{O}\mathcal{H}} = \frac{h}{r^2} \boldsymbol{k}} \tag{3.9}$$

Note that the angular velocity of the intermediate frame with respect to the inertial frame lies in the orbital frame.

Two important conclusions are inferred from this result: i) in absence of perturbation the orbital plane remain fixed on the inertial frame and ii) in a perturbed environment the motion of the orbital plane is described by pure rotation about the radius vector of the particle.

3.3.3 Ideal frame concept

In absence of perturbations ($\boldsymbol{a}_p = 0$) the intermediate frame does not move in the inertial space. That is the reason that lead Hansen (Ref.[22]) to call this frame the $ideal frame$, defined by the basis $\mathcal{H} : \lfloor \boldsymbol{i}_H, \boldsymbol{j}_H, \boldsymbol{k}_H \rceil$. In a perturbed environment, the ideal frame is no longer fixed, but moving, and its instantaneous position may be viewed as the image of the ideal frame at epoch by a finite rotation.

There exists two main ideal frames. The first one is defined by the initial condition $(\boldsymbol{r_0}, \boldsymbol{v_0})$. This one is used in DEPRIT method, and it is called the $departure frame$. The second one coincides with the *perifocal frame* at the initial time, which is used in DROMO method and it is called the *departure perifocal frame*. Both frames share the same angular velocity.

Note that the $\boldsymbol{k}$ versor of the orbital frame coincides with the $\boldsymbol{k}_H$ versor of the ideal frame ($\boldsymbol{k} = \boldsymbol{k}_H$). Hence, the versors of the orbital frame can be expressed as a function of the versors of the ideal frame by the angle θ in DEPRIT, and the angle σ in DROMO.

3.3.4 Euler parameters

Two main frames are involved in the movement of the particle: (1) the fixed frame $\mathcal{I}$ whose unitary versors are $[\boldsymbol{i}_I, \boldsymbol{j}_I, \boldsymbol{k}_I]$, and (2) the ideal frame $\mathcal{H}$ whose normalized versors are $[\boldsymbol{i}_H, \boldsymbol{j}_H, \boldsymbol{k}_H]$. These set of vectors are related by a rotation represented by the following equation:

$$[\boldsymbol{i}_H, \boldsymbol{j}_H, \boldsymbol{k}_H] = [\boldsymbol{i}_I, \boldsymbol{j}_I, \boldsymbol{k}_I]Q(t) \tag{3.10}$$

where $Q(t)$ is a third order orthogonal matrix which is time dependant. Only three of nine elements are independent. Hence, the minimum number of elements to eliminate the previous singularity is four. For this aim, Euler parameters are selected $\epsilon = (\eta + (\epsilon_1, \epsilon_2, \epsilon_3))$, and the matrix is then given by:

$$Q = \begin{bmatrix} 1-2(\epsilon_2^2+\epsilon_3^2) & 2\epsilon_1\epsilon_2 - 2\eta\epsilon_3 & 2\epsilon_1\epsilon_3 + 2\eta\epsilon_2 \\ 2\epsilon_1\epsilon_2 + 2\eta\epsilon_3 & 1-2(\epsilon_1^2+\epsilon_3^2) & 2\epsilon_2\epsilon_3 - 2\eta\epsilon_1 \\ 2\epsilon_1\epsilon_3 - 2\eta\epsilon_2 & 2\epsilon_2\epsilon_3 + 2\eta\epsilon_1 & 1-2(\epsilon_1^2+\epsilon_2^2) \end{bmatrix} \tag{3.11}$$

3.4 Deprit method

Three reference frames should be defined. First, the inertial reference frame is denoted by $(\mathcal{I})$. Second, the ideal frame $\mathcal{H}'$ defined by the basis $\mathcal{H}' : \lfloor \boldsymbol{i}_{H'}, \boldsymbol{j}_{H'}, \boldsymbol{k}_{H'} \rceil$, where the versors $(\boldsymbol{i}_H, \boldsymbol{j}_{H'})$ are contained in the orbital plane which contains the osculating ellipse. This frame coincides with the orbital frame at epoch an it is called *the departure frame* as seen in the previous section. Third, the orbital frame $(\mathcal{O})$ rotates attached to the particle. The orbital frame is defined by the basis $\mathcal{O} : \lfloor \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k} \rceil$ such that: versor $\boldsymbol{i}$ is parallel to the radius-vector pointing zenithwards; versor $\boldsymbol{k}$ is described by the angular momentum vector, $\boldsymbol{h}$; and $\boldsymbol{j}$ completes an orthogonal dextral reference frame. That is:

$$\boldsymbol{i} = \frac{\boldsymbol{r}}{r}, \qquad \boldsymbol{k} = \frac{\boldsymbol{h}}{h}, \qquad \boldsymbol{j} = \boldsymbol{k} \times \boldsymbol{i} \tag{3.12}$$

DEPRIT introduced a new set of non-classical elements $[\boldsymbol{\epsilon}, h, C, S, F]$ to describe the osculating orbit and they are termed ideal because they refer to an ideal frame of reference. These elements appears naturally when formulating the motion os the particle: the triple (h, C, S) fully characterize the orbit geometry, whereas $(\eta, \epsilon_1, \epsilon_2, \epsilon_3)$ are the Euler parameters associated to the quaternion $\boldsymbol{\epsilon}$. This quaternion defines the orientation of the ideal frame with respect to the inertial reference.

3.4.1 Orientation of the ideal frame

The first four elements are the Euler parameters $(\epsilon_1, \epsilon_2, \epsilon_3, \eta)$. They represent a finite rotation, which are free of any kind of singularities, and provide the instantaneous position of the ideal frame $\mathcal{O}$. They are related with the angular velocity $\omega_{\mathcal{H}'\mathcal{I}}$ by the following equations:

$$\frac{\mathrm{d}\epsilon}{\mathrm{d}\,t} = \frac{1}{2}(\epsilon \times \boldsymbol{\omega}_{\mathcal{H}'\mathcal{I}} + \eta\boldsymbol{\omega}_{\mathcal{H}'\mathcal{I}}), \qquad \frac{\mathrm{d}\eta}{\mathrm{d}\,t} = -\frac{1}{2}\epsilon \cdot \boldsymbol{\omega}_{\mathcal{H}'\mathcal{I}} \tag{3.13}$$

Let θ be the angle between the radius vector pointing the particle and the direction $\boldsymbol{i}_{H'}$ of the ideal frame. This direction is named the *departure point* (Fig.(3.2)). The orbital basis can be expressed in terms of the ideal basis by the angle θ as follows:

$$\boldsymbol{i} = \cos\theta \boldsymbol{i}_{H'} + \sin\theta \boldsymbol{j}_{H'}$$

$$\boldsymbol{j} = -\sin\theta \boldsymbol{i}_{H'} + \cos\theta \boldsymbol{j}_{H'} \tag{3.14}$$

$$\boldsymbol{k} = \boldsymbol{k}_{H'} \tag{3.15}$$

Hence, considering the definition of the angular velocity given by Eq.(3.7), these equations translate into a system of differential equations in terms of the perturbing force.

$$\boxed{\begin{aligned}
\frac{\mathrm{d}\epsilon_1}{\mathrm{d}\,\tau} &= \frac{1}{2}a_{pz}\frac{r}{h}(\eta\cos\theta - \epsilon_3\sin\theta) \\
\frac{\mathrm{d}\epsilon_2}{\mathrm{d}\,\tau} &= \frac{1}{2}a_{pz}\frac{r}{h}(\eta\sin\theta + \epsilon_3\cos\theta) \\
\frac{\mathrm{d}\epsilon_3}{\mathrm{d}\,\tau} &= \frac{1}{2}a_{pz}\frac{r}{h}(\epsilon_1\sin\theta - \epsilon_2\cos\theta) \\
\frac{\mathrm{d}\eta}{\mathrm{d}\,\tau} &= \frac{1}{2}a_{pz}\frac{r}{h}(-\epsilon_1\cos\theta - \epsilon_2\sin\theta)
\end{aligned}} \tag{3.16}$$

Where it is recommended to use this notation

$$p = \frac{r}{h}\cos\theta, \qquad q = \frac{r}{h}\sin\theta$$

Moreover, these parameters must satisfy the identity

$$\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \eta^2 = 1$$

.

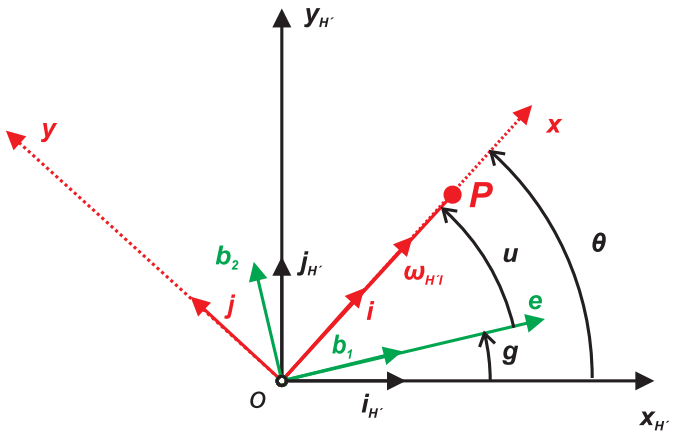


FIGURE 3.2: Ideal frame in Deprit method

3.4.2 Orbit geometry

Let $\boldsymbol{h}$ be the vector of angular momentum per unit of mass,whose time evolution has been already obtained in Eq.(3.3). Given that $h^2 = \boldsymbol{h} \cdot \boldsymbol{h}$ we obtain the variation of its module $h = |\boldsymbol{h}|$

$$\boxed{\frac{\mathrm{d}h}{\mathrm{d}\tau} = r(\boldsymbol{a}_p \cdot \boldsymbol{j})} \tag{3.17}$$

In order to set the osculating ellipse on the orbital plane, Deprit used the parameters C and S, which are defined as:

$$C = \frac{e\cos g}{h}; \qquad S = \frac{e\sin g}{h}$$

where e represents the eccentricity, and g is the angle between the eccentricity vector $\boldsymbol{e}$ and the departure point $\boldsymbol{i}_{H'}$. So that C and S represent the projection of $\boldsymbol{e}$ in the ideal frame $\boldsymbol{e} = hC\boldsymbol{i}_{H'} + hS\boldsymbol{j}_{H'}$. Taking this equation together with the hodograph equation $\boldsymbol{v} = \frac{1}{h}(\boldsymbol{j} + \boldsymbol{k} \times \boldsymbol{e})$ we infer that

$$-\frac{\boldsymbol{j}}{h} + \boldsymbol{v} = C\boldsymbol{j}_{H'} + S\boldsymbol{i}_{H'}$$

and differentiating this equation with respect the ideal frame and considering Eq.(3.4) and (3.1), it is obtained that

$$(1 + \frac{r}{h^2})(\boldsymbol{a}_p \cdot \boldsymbol{j})\boldsymbol{j} + (\boldsymbol{a}_p \cdot \boldsymbol{i})\boldsymbol{i} = \frac{\mathrm{d}C}{\mathrm{d}\tau}\boldsymbol{j}_{H'} - \frac{\mathrm{d}S}{\mathrm{d}\tau}\boldsymbol{i}_{H'}$$

Naming the left-hand member of this equation as the effective perturbing force $\boldsymbol{a}_f$, two simple differential equations are finally obtained

$$\boxed{\dot{C} = \boldsymbol{a}_f \cdot \boldsymbol{j}_{H'}} \qquad \boxed{\dot{S} = -\boldsymbol{a}_f \cdot \boldsymbol{i}_{H'}} \tag{3.18}$$

3.4.3 Timing in the osculating ellipse

Finally, the timing on the osculating ellipse must be determined. Let consider the classical Kepler equation and its time derivative, where M is the mean anomaly and E the eccentric anomaly, both reckoned from the eccentricity vector $\boldsymbol{e}$.

$$M = E - e\sin E \longrightarrow \dot{M} = \dot{E} - \frac{\mathrm{d}}{\mathrm{d}\,\tau}(e\sin E)$$

Introducing in this equation the definition of F as the mean anomaly reckoned form the departure point ($F = M + g$). After several combinations it is obtained that

$$\dot{F} = \sqrt{\frac{1}{a^3}} + \frac{he}{1+\eta}(\boldsymbol{a}_p \cdot \frac{\boldsymbol{e}}{e}) - 2\frac{r}{\sqrt{a}}(\boldsymbol{a}_p \cdot \boldsymbol{i})$$

where η is defined as $\eta = \sqrt{1-e^2}$ and a represents the value of the semi-mayor axis. In order to avoid having to determine the eccentricity vector, we decompose the right-hand member in the instantaneous ideal frame

$$\boxed{\dot{F} = \sqrt{\frac{1}{a^3}} + \frac{h^2}{1+\eta}(C\dot{S} - \dot{C}S) + 2\frac{r}{h}\eta(\dot{S}\cos\theta - \dot{C}\sin\theta)} \tag{3.19}$$

These Eqs.(3.16)-(3.19) represent the differential system of the Deprit formulation for the Keplerian problem.

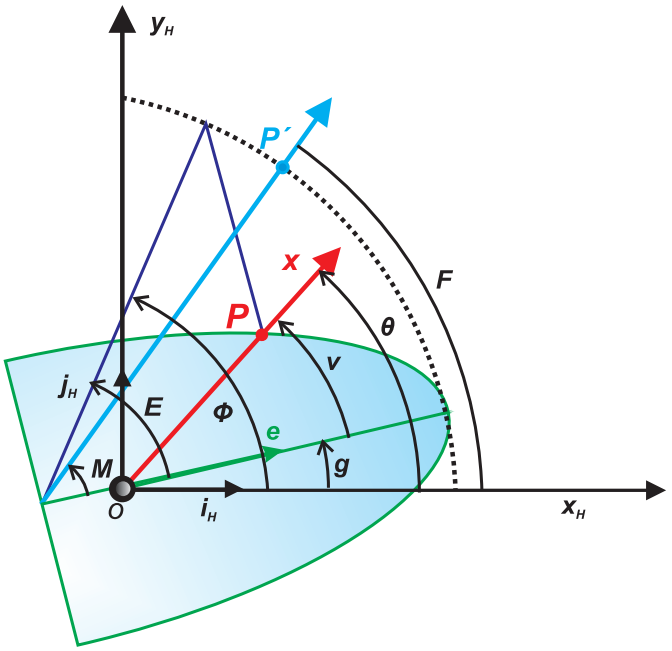


FIGURE 3.3: Mean anomalies representation in Deprit method

3.4.4 From Deprit variables to state vector

In order to compute these equations and to obtain the final state vector, it is necessary to transform the classical orbital elements to Deprit elements and vice-versa.

Let us assume that the dynamical state of the system is known in terms of the DEPRIT variables $[\boldsymbol{\epsilon}, h, C, S, F]$ and we aim to obtain the classical state vector given by $[\hat{\boldsymbol{r}}, \hat{\boldsymbol{v}}]$ in dimensional variables. Obviously, the physical time t is determined from the non-dimensional time.

$$t - t_o = \frac{\tau}{\omega_c} \tag{3.20}$$

First fo all, it is obtain the ideal frame as a function of the quaternion $\boldsymbol{\epsilon}$ obtained during the integration

$$[\boldsymbol{i}_{H'}, \boldsymbol{j}_{H'}, \boldsymbol{k}_{H'}] = [\boldsymbol{i}_I, \boldsymbol{j}_I, \boldsymbol{k}_I] Q(\boldsymbol{\epsilon})$$

Where Q is defined in Eq.3.11.

We define e_x and e_y as the projections of the eccentricity vector $\boldsymbol{e}$ in the ideal frame

$$e_x = hC = e\cos g, \qquad e_y = hS = e\sin g$$

Let M and E be the mean anomaly and eccentric anomaly respectively, reckoned in the classical perifocal frame. Let F be the mean anomaly, and ϕ the eccentric anomaly reckoned in the ideal frame. Hence, the classical Kepler equation in the perifocal frame is converted to

$$F = M + g, \qquad \phi = E + g \longrightarrow M = E - e\sin E \longrightarrow F = \phi - (e_x \sin\phi - e_y \cos\phi) \longrightarrow \phi$$

From this equation ϕ it is obtained, and it is necessary to solve Kepler equation. Normally a Newton-Raphson algorithm permits to solve it with acceptable. It is a disadvantage for this method, given that finding its solution with the current methods, can be time-consuming and can provoke loss of precision, in contrast to DROMO, where this equation do not appear.
In the same way, we transform the classical identities involving anomalies reckoned in the line of apsides, where ν represents the true anomaly in the perifocal frame, and θ in the ideal frame.

$$\left\{\begin{array}{rcl} r & = & a(1-e\cos E) \\ \cos\nu & = & \dfrac{a}{r}(\cos E - e) \\ \sin\nu & = & \dfrac{a}{r}\eta\sin E \end{array}\right\} \longrightarrow \left\{\begin{array}{rcl} r & = & a(1-e_x\cos\phi - e_y\sin\phi) \\ \cos\theta & = & \dfrac{a}{r}(\cos\phi - e_x + \dfrac{\phi-F}{1+\eta}e_y) \\ \sin\theta & = & \dfrac{a}{r}(\sin\phi - e_y - \dfrac{\phi-F}{1+\eta}e_x) \end{array}\right\}$$

Where a is the semi-major axis of the ellipse ans η is a parameter related to the eccentricity defined by

$$a = \frac{h^2}{\eta^2}, \qquad \eta = (1 - e_x^2 - e_y^2)$$

Being r and θ known, we are able to obtain the position vector of the particle as follows

$$\boldsymbol{r} = r(\cos\theta \boldsymbol{i}_{H'} + \sin\theta \boldsymbol{j}_{H'})$$

We can easily obtain the velocity vector immediately from the hodograph equation using the next relations

$$\boldsymbol{e} = e_x \boldsymbol{i}_{H'} + e_y \boldsymbol{j}_{H'}, \qquad \boldsymbol{j} = -\sin\theta \boldsymbol{i}_{H'} + \cos\theta \boldsymbol{j}_{H'}, \qquad \longrightarrow \boldsymbol{v} = \frac{1}{h}(\boldsymbol{k}\times\boldsymbol{e} + \boldsymbol{j})$$

It has been solved the problem of determinate the non-dimensional position and velocity vectors. Finally, we have to correctly turn the variables dimensional.

$$\hat{\boldsymbol{r}} = L_c \boldsymbol{r} \tag{3.21}$$

$$\hat{\boldsymbol{v}} = \omega_c L_c \boldsymbol{v} \tag{3.22}$$

3.4.5 Initial conditions

Let us assume that we know the state vector of the particle at epoch in the inertial frame (in the non-dimensional form) $(\boldsymbol{r}_0, \boldsymbol{v}_0)$. Given this conditions it si fairly straightforward to calculate the angular momentum and eccentricity vector

$$\boldsymbol{h}_0 = \boldsymbol{r}_0 \times \boldsymbol{v}_0, \qquad \boldsymbol{e}_0 = -\frac{\boldsymbol{r}_0}{|\boldsymbol{r}_0|} - \boldsymbol{h}_0 \times \boldsymbol{v}_0 \tag{3.23}$$

From this equations we can trivially obtain the unitary vectors of the *orbital frame*

$$\boldsymbol{i}(0) = \frac{\boldsymbol{r}_0}{|\boldsymbol{r}_0|}, \qquad , \boldsymbol{k}(0) = \frac{\boldsymbol{h}_0}{|\boldsymbol{h}_0|}, \qquad \boldsymbol{j}(0) = \boldsymbol{k}(0) \times \boldsymbol{i}(0) \tag{3.24}$$

In the initial state the orbital frame coincides with the *ideal frame*

$$\boldsymbol{i}_{H'}(0) = \boldsymbol{i}(0), \qquad \boldsymbol{j}_{H'}(0) = \boldsymbol{j}(0), \qquad \boldsymbol{k}_{H'}(0) = \boldsymbol{k}(0) \tag{3.25}$$

With the versors of the ideal frame we can obtain de matrix Eq.(3.11) and it is also straightforward to calculate the initial value of the Euler parameters.

The initial value of the mean anomaly $F(0)$ can be obtain as a sum of $F_0 = g_0 + M_0$. The angle g_0 is obtained as the angle between the eccentricity vector and the position vector

$$\cos g_0 = \boldsymbol{i}_{H'}(0) \cdot \frac{\boldsymbol{e}_0}{|\boldsymbol{e}_0|}, \qquad \sin g_0 = \boldsymbol{j}_{H'}(0) \cdot \frac{\boldsymbol{e}_0}{|\boldsymbol{e}_0|} \tag{3.26}$$

From this angle we can also obtain the C and S parameters as follows

$$C(0) = |\boldsymbol{e}_0| \cos g_0, \qquad S(0) = |\boldsymbol{e}_0| \sin g_0 \tag{3.27}$$

Now we have to calculate the initial anomalies reckoned from the eccentricity vector, namely the true anomaly ν_0 and the eccentric anomaly E_0.

$$\cos \nu_0 = \frac{\boldsymbol{e}_0 \cdot \boldsymbol{r}_0}{|\boldsymbol{e}_0||\boldsymbol{r}_0|}, \qquad \sin \nu_0 = \frac{\boldsymbol{k}(0) \cdot (\boldsymbol{e}_0 \times \boldsymbol{r}_0)}{|\boldsymbol{e}_0||\boldsymbol{r}_0|} \tag{3.28}$$

$$\cos E_0 = \frac{|\boldsymbol{e}_0| + \cos \nu_0}{1 + |\boldsymbol{e}_0| \cos \nu_0}, \qquad \sin E_0 = \frac{\sqrt{1 - |\boldsymbol{e}_0|^2} \sin \nu_0}{1 + |\boldsymbol{e}_0| \cos \nu_0} \tag{3.29}$$

From this equations we get the mean anomaly M_0

$$M_0 = E_0 - |\boldsymbol{e}_0| \sin E_0 \tag{3.30}$$

Finally, the initial values of the DEPRIT variables are:

$$C(0) = C_0, \qquad S(0) = S_0, \qquad h(0) = h_0, \qquad F(0) = M_0 + g_0 \tag{3.31}$$

$$\eta(0) = \eta_0, \qquad \epsilon_1(0) = \epsilon_{10}, \qquad \epsilon_2(0) = \epsilon_{20}, \qquad \epsilon_3(0) = \epsilon_{30} \tag{3.32}$$

3.4.6 Summary of equations

ORBITAL PLANE ORIENTATION
$$\frac{d\epsilon_1}{d\tau} = \frac{1}{2}a_{pz}\frac{r}{h}(\eta\cos\theta - \epsilon_3\sin\theta)$$
$$\frac{d\epsilon_2}{d\tau} = \frac{1}{2}a_{pz}\frac{r}{h}(\eta\sin\theta + \epsilon_3\cos\theta)$$
$$\frac{d\epsilon_3}{d\tau} = \frac{1}{2}a_{pz}\frac{r}{h}(\epsilon_1\sin\theta - \epsilon_2\cos\theta)$$
$$\frac{d\eta}{d\tau} = \frac{1}{2}a_{pz}\frac{r}{h}(-\epsilon_1\cos\theta - \epsilon_2\sin\theta)$$
$$\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \eta^2 = 1$$
ELLIPSE IN THE ORBITAL PLANE
$$\dot{h} = ra_{py}$$ $$\dot{C} = \boldsymbol{a}_f \cdot \boldsymbol{j}_{\boldsymbol{H'}}$$ $$\dot{S} = -\boldsymbol{a}_f \cdot \boldsymbol{i}_{\boldsymbol{H'}}$$
TIMING IN THE OSCULATING ELLIPSE
$$\dot{F} = n + \frac{h^2}{1+\eta}(C\dot{S} - \dot{C}S) + 2\eta(p\dot{S} - q\dot{C})$$

TABLE 3.1: Summary of equations for Deprit method

3.5 DROMO method

Three reference frames should be defined. First, the inertial frame is denoted by $(\mathcal{I})$. Second, the orbital frame $(\mathcal{O})$ rotates attached to the particle. The orbital frame is defined by the basis $\mathcal{O} : \lfloor \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k} \rceil$ such that: versor $\boldsymbol{i}$ is parallel to the radius-vector pointing zenithwards; versor $\boldsymbol{k}$ is described by the angular momentum vector, $\boldsymbol{h}$; and $\boldsymbol{j}$ completes an orthogonal dextral reference frame. That is:

$$\boldsymbol{i} = \frac{\boldsymbol{r}}{r}, \qquad \boldsymbol{k} = \frac{\boldsymbol{h}}{h}, \qquad \boldsymbol{j} = \boldsymbol{k} \times \boldsymbol{i} \tag{3.33}$$

Finally, the *ideal perifocal frame* $(\mathcal{H})$ is introduced. This reference is fixed to the orbital plane, describe its dynamics and coincides with the perifocal frame at epoch. It was first introduced by Hansen in Ref.[22] and further developed by Deprit in Ref.[21], the term *ideal* refers to a particular intermediate frame: on this frame the motion is reducible to the rotation of frame $(\mathcal{O})$ referred to the orbital plane $(\mathcal{H})$, described by the angular momentum, and the motion of the orbital frame, and the motion of the orbital frame $(\mathcal{H})$ with respect to the inertial reference $(\mathcal{I})$,due to perturbations. In particular, the ideal perifocal frame corresponds to the departure perifocal frame at epoch.

The independent variable in DROMO is the ideal anomaly, σ. If defines the angular evolution of the radius-vector referred to the departure perifocal frame. If external perturbations exists, the instantaneous perifocal frame may differ from the departure perifocal frame. Let β be the angle that describe the motion. The ideal anomaly is therefore related to the true anomaly ν, through the expression:

$$\sigma = \beta + \nu \tag{3.34}$$

DROMO introduced a new set of non-classical elements $(\zeta_1, \zeta_2, \zeta_3, \eta, \epsilon_1, \epsilon_2, \epsilon_3)$ to describe the osculating orbit. These elements appears naturally when formulating the motion os the particle: the triple $(\zeta_1, \zeta_2, \zeta_3)$ fully characterize the orbit geometry, whereas $(\eta, \epsilon_1, \epsilon_2, \epsilon_3)$ are the Euler parameters associated to the quaternion $\boldsymbol{\epsilon}$. This quaternion defines the orientation of the perifocal frame with respect to the inertial reference. For further information of this method see reference [11].

3.5.1 Orientation of the ideal frame

The first four elements are the Euler parameters $(\epsilon_1, \epsilon_2, \epsilon_3, \eta)$. They represent a finite rotation, which are free of any kind of singularities, and provide the instantaneous position of the ideal frame $\mathcal{O}$. They are related with the angular velocity $\omega_{\mathcal{HI}}$ by the following equations:

$$\frac{\mathrm{d}\epsilon}{\mathrm{d}\,t} = \frac{1}{2}(\epsilon \times \boldsymbol{\omega}_{\mathcal{HI}} + \eta\boldsymbol{\omega}_{\mathcal{I}}), \qquad \frac{\mathrm{d}\eta}{\mathrm{d}\,t} = -\frac{1}{2}\epsilon \cdot \boldsymbol{\omega}_{\mathcal{HI}} \tag{3.35}$$

Let σ be the angle between the radius vector pointing the particle and the direction $\boldsymbol{i}_H$ of the ideal perifocal frame. This direction is named the *departure perifocal point* (Fig.(3.4)). The orbital basis can be expressed in terms of the ideal basis by the angle σ as follows:

$$\boldsymbol{i} = \cos\sigma \boldsymbol{i}_H + \sin\sigma \boldsymbol{j}_H \tag{3.36}$$

$$\boldsymbol{j} = -\sin\sigma \boldsymbol{i}_H + \cos\sigma \boldsymbol{j}_H \tag{3.37}$$

$$\boldsymbol{k} = \boldsymbol{k}_H \tag{3.38}$$

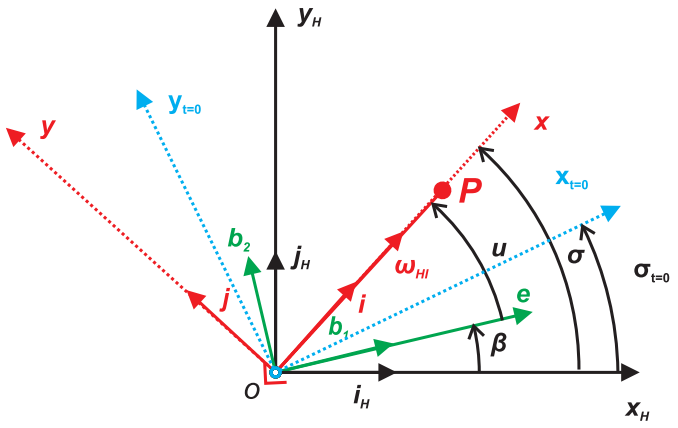


FIGURE 3.4: Ideal frame in Dromo method

Hence, considering the definition of the angular velocity given by Eq.(3.7), these equations translate into a system of differential equations in terms of the perturbing force.

$$\boxed{\begin{aligned}
\frac{\mathrm{d}\epsilon_1}{\mathrm{d}\,\tau} &= \frac{1}{2}a_{pz}\frac{r}{h}(\eta\cos\sigma - \epsilon_3\sin\sigma) \\
\frac{\mathrm{d}\epsilon_2}{\mathrm{d}\,\tau} &= \frac{1}{2}a_{pz}\frac{r}{h}(\eta\sin\sigma + \epsilon_3\cos\sigma) \\
\frac{\mathrm{d}\epsilon_3}{\mathrm{d}\,\tau} &= \frac{1}{2}a_{pz}\frac{r}{h}(\epsilon_1\sin\sigma - \epsilon_2\cos\sigma) \\
\frac{\mathrm{d}\eta}{\mathrm{d}\,\tau} &= \frac{1}{2}a_{pz}\frac{r}{h}(-\epsilon_1\cos\sigma - \epsilon_2\sin\sigma)
\end{aligned}}$$

Moreover, these parameters must satisfy the identity

$$\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \eta^2 = 1.$$

3.5.2 Orbit geometry

Let $\boldsymbol{h}$ be the vector of angular momentum per unit of mass,whose time evolution has been already obtained in Eq.(3.3). Given that $h^2 = \boldsymbol{h} \cdot \boldsymbol{h}$ we obtain the variation of its module $h = |\boldsymbol{h}|$

$$\frac{\mathrm{d}\boldsymbol{h}}{\mathrm{d}\,\tau} = r(\boldsymbol{a}_p \cdot \boldsymbol{j})\boldsymbol{k} \tag{3.39}$$

Given that σ is the angle between the position vector and the departure perifocal point, it stands that

$$\boldsymbol{\omega}_{\mathcal{OH}} = \dot{\sigma}\boldsymbol{k}$$

It is easy to obtain the time evolution of the true anomaly σ

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\,\tau} = \frac{h}{r^2} \tag{3.40}$$

Let consider the new DROMO variable ζ_3 as the inverse of the magnitude of angular momentum $h = |\boldsymbol{h}|$:

$$\zeta_3 = \frac{1}{h} \tag{3.41}$$

From this relation we obtain

$$\boxed{\frac{\mathrm{d}\zeta_3}{\mathrm{d}\,\sigma} = -r^3\zeta_3^3 a_{py}} \tag{3.42}$$

and from the equation 3.40 we get

$$\boxed{\frac{\mathrm{d}\tau}{\mathrm{d}\,\sigma} = r^2\zeta_3} \tag{3.43}$$

We can obtain the σ-derivate of the velocity vector with respect to the ideal frame, using the Coriolis theorem and Eq.(3.1) and Eq.(3.40) as

$$\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}\,\sigma}\frac{1}{h}(-\boldsymbol{i} + r^2\boldsymbol{a}_{p||}) \tag{3.44}$$

where $a_{p||}$ is the component of $\boldsymbol{a}_p$ parallel to the orbital plane.

Let consider the non-dimensional eccentricity vector $\boldsymbol{e} = -\boldsymbol{i} - \boldsymbol{h} \times \boldsymbol{v}$. Differentiating this equation with respect to σ in the ideal frame and using Eq.(3.44), Eq.(3.39) and Eq.(3.4) we get

$$\frac{\mathrm{d}\boldsymbol{e}}{\mathrm{d}\,\sigma} = \frac{h^2}{h^2}(a_{py}[\boldsymbol{i} + \boldsymbol{e}] - \frac{h^2}{r}\boldsymbol{k} \times \boldsymbol{a}_{p||} \tag{3.45}$$

Let define ζ_1 and ζ_2 as the coordinates of the eccentricity vector in the ideal frame $\boldsymbol{e} = \zeta_1 \boldsymbol{i}_H + \zeta_2 \boldsymbol{j}_H$. Therefore, the σ-derivate of the eccentricity vector with respect to the ideal frame will be

$$\frac{\mathrm{d}\boldsymbol{e}}{\mathrm{d}\,\sigma} = \frac{\mathrm{d}\zeta_1}{\mathrm{d}\,\sigma}\boldsymbol{i}_H + \frac{\mathrm{d}\zeta_2}{\mathrm{d}\,\sigma}\boldsymbol{j}_H$$

Comparing this equation with the Eq.(3.45) projected in the perifocal frame, we finally obtain that

$$\boxed{\frac{\mathrm{d}\zeta_1}{\mathrm{d}\,\sigma} = +\boldsymbol{a}_f \cdot \boldsymbol{j}_H + \frac{r^3}{h^2}a_{py}\zeta_1} \qquad \boxed{\frac{\mathrm{d}\zeta_2}{\mathrm{d}\,\sigma} = -\boldsymbol{a}_f \cdot \boldsymbol{i}_H + \frac{r^3}{h^2}a_{py}\zeta_2} \tag{3.46}$$

where a_f is the effective perturbing force for the DROMO method. It can be defined as

$$a_f = r^2[(1 + \frac{r}{h^2})(\boldsymbol{a}_p \cdot \boldsymbol{j})\boldsymbol{j} + (\boldsymbol{a}_p \cdot \boldsymbol{i})\boldsymbol{i}] \tag{3.47}$$

and satisfy $a_f = r^2 a_{f'}$, where $a_{f'}$ is the effective perturbing force in Deprit method.

In summary, in terms of the DROMO variables $(\sigma; \tau, \zeta_1, \zeta_2, \zeta_3)$ the motion of the particle M is governed by the previous boxed equations- The right hand members of these equations should be expressed in terms of $(\sigma; \tau, \zeta_1, \zeta_2, \zeta_3)$, as we show in the following section.

3.5.3 From DROMO variables to state vector

Let us assume that the dynamical state of the system is known in terms of the DROMO variables $[\boldsymbol{\epsilon}, \zeta_1, \zeta_2, \zeta_3, \tau, \sigma]$ and we aim to obtain the classical state vector given by $[\hat{\boldsymbol{r}}, \hat{\boldsymbol{v}}]$ in dimensional variables. Obviously, the physical time t is determined from the non-dimensional time.

$$t - t_o = \frac{\tau}{\omega_c}$$

First fo all, it is obtain the ideal frame as a function of the quaternion $\boldsymbol{\epsilon}$ obtained during the integration

$$[\boldsymbol{i}_H, \boldsymbol{j}_H, \boldsymbol{k}_H] = [\boldsymbol{i}_I, \boldsymbol{j}_I, \boldsymbol{k}_I]Q(\boldsymbol{\epsilon})$$

Where Q is defined in Eq.(3.11).

The position vector can be expressed in the departure perifocal frame as

$$\boldsymbol{r} = r(\cos\sigma \boldsymbol{i}_H + \sin\sigma \boldsymbol{j}_H) = \frac{1}{\zeta_3^2 s}(\cos\sigma \boldsymbol{i}_H + \sin\sigma \boldsymbol{j}_H)$$

From the hodograph equation, vector $\boldsymbol{v}$, can be expressed in the departure perifocal frame as

$$\boldsymbol{v} = \zeta_3 \boldsymbol{j} + \zeta_1 \boldsymbol{j}_H - \zeta_2 \boldsymbol{i}_H = \zeta_3 -(\sin\sigma + \zeta_2)\boldsymbol{i}_H + (\cos\sigma + \zeta_1)\boldsymbol{j}_H$$

It has been solved the problem of determinate the non-dimensional position and velocity vectors. Finally, we have to correctly turn the variables dimensional.

$$\hat{\boldsymbol{r}} = L_c \boldsymbol{r}$$
$$\hat{\boldsymbol{v}} = \omega_c L_c \boldsymbol{v}$$

3.5.4 Additional simplifications

Given that (r, σ) are the polar coordinates of M in the departure perifocal frame $\mathcal{P}$: as a consequence the velocity with respect to this frame turns out to be

$$\boldsymbol{v} = \frac{\mathrm{d}r}{\mathrm{d}\tau}\boldsymbol{i}_H + \frac{h}{r}\boldsymbol{j}_H \tag{3.48}$$

Taking into account this equation and the definition of the eccentricity vector $\boldsymbol{e} = -\boldsymbol{i} - \boldsymbol{h} \times \boldsymbol{v}$, it can be expressed as:

$$\boldsymbol{e} = (\frac{h^2}{r} - 1)\boldsymbol{i}_H - h\frac{\mathrm{d}r}{\mathrm{d}\tau}\boldsymbol{j}_H \tag{3.49}$$

Considering the representation $\frac{\mathrm{d}\boldsymbol{e}}{\mathrm{d}\sigma} = \frac{\mathrm{d}\zeta_1}{\mathrm{d}\sigma}\boldsymbol{i}_H + \frac{\mathrm{d}\zeta_2}{\mathrm{d}\sigma}\boldsymbol{j}_H$ and using Eq.(3.36) we finally obtain

$$\zeta_1 = (\frac{h^2}{r} - 1)\cos\sigma + h\frac{\mathrm{d}r}{\mathrm{d}\tau}\sin\sigma \tag{3.50}$$
$$\zeta_2 = (\frac{h^2}{r} - 1)\sin\sigma - h\frac{\mathrm{d}r}{\mathrm{d}\tau}\cos\sigma \tag{3.51}$$

The inverse relation of these equations are

$$\frac{1}{r} = \zeta_3^2 s \tag{3.52}$$
$$\frac{\mathrm{d}r}{\mathrm{d}\tau} = \zeta_3(\zeta_1 \sin\sigma - \zeta_2 \cos\sigma) \tag{3.53}$$

where $s = 1+\zeta_1 \cos\sigma - \sin\sigma$ is a parameter introduced in order to simplify the equations. We already have all the right-hand members in terms of DROMO variables.

3.5.5 Initial conditions

Let us assume that we know the state vector of the particle at epoch in the inertial frame (in the non-dimensional form) $(\boldsymbol{r}_0, \boldsymbol{v}_0)$. Given this conditions it is fairly straightforward to calculate the angular momentum and eccentricity vector

$$\boldsymbol{h}_0 = \boldsymbol{r}_0 \times \boldsymbol{v}_0, \qquad \boldsymbol{e}_0 = -\frac{\boldsymbol{r}_0}{|\boldsymbol{r}_0|} - \boldsymbol{h}_0 \times \boldsymbol{v}_0 \tag{3.54}$$

which permit to calculate the versors of the *perifocal departure frame* as follows

$$\boldsymbol{k}_H(0) = \frac{\boldsymbol{h}_0}{|\boldsymbol{h}_0|}, \qquad \boldsymbol{i}_H(0) = \frac{\boldsymbol{e}_0}{|\boldsymbol{e}_0|}, \qquad \boldsymbol{j}_H(0) = \boldsymbol{k}_H(0) \times \boldsymbol{i}_H(0) \tag{3.55}$$

and also we can trivially obtain the unitary vectors of the *orbital frame*

$$\boldsymbol{i}(0) = \frac{\boldsymbol{r}_0}{|\boldsymbol{r}_0|}, \qquad , \boldsymbol{k}(0) = \frac{\boldsymbol{h}_0}{|\boldsymbol{h}_0|}, \qquad \boldsymbol{j}(0) = \boldsymbol{k}(0) \times \boldsymbol{i}(0) \tag{3.56}$$

With the versors of the perifocal frame we can obtain de matrix Eq.(3.11) and it is also straightforward to calculate the initial value of the Euler parameters.

The initial value of the true anomaly is defined by the angle between the position vector and the eccentricity one

$$\cos\sigma_0 = \boldsymbol{i}_H(0) \cdot \boldsymbol{i}(0), \qquad \sin\sigma_0 = \boldsymbol{i}_H(0) \cdot \boldsymbol{j}(0) \tag{3.57}$$

Finally, the initial values of the DROMO variables are:

$$\begin{aligned} &\sigma(0) = \sigma_0, \quad \tau(0) = 0, \quad \zeta_1(0) = e_0, \quad \zeta_2(0) = 0, \quad \zeta_3(0) = 1/h_0 \\ &\eta(0) = \eta_0, \quad \epsilon_1(0) = \epsilon_{10}, \quad \epsilon_2(0) = \epsilon_{20}, \quad \epsilon_3(0) = \epsilon_{30} \end{aligned} \tag{3.58}$$

3.5.6 Summary of equations

ORBITAL PLANE ORIENTATION
$$\frac{d\epsilon_1}{d\sigma} = \frac{1}{2}\tilde{a}_{pz}(\eta\cos\sigma - \epsilon_3\sin\sigma)$$
$$\frac{d\epsilon_2}{d\sigma} = \frac{1}{2}\tilde{a}_{pz}(\eta\sin\sigma + \epsilon_3\cos\sigma)$$
$$\frac{d\epsilon_3}{d\sigma} = \frac{1}{2}\tilde{a}_{pz}(\epsilon_1\sin\sigma - \epsilon_2\cos\sigma)$$
$$\frac{d\eta}{d\sigma} = \frac{1}{2}\tilde{a}_{pz}(-\epsilon_1\cos\sigma - \epsilon_2\sin\sigma)$$
$$\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \eta^2 = 1$$
ORBIT GEOMETRY
$$\frac{d\zeta_1}{d\sigma} = s\sin\sigma(\tilde{a}_{px}) + (\zeta_1 + (1+s)\cos\sigma)(\tilde{a}_{py})$$
$$\frac{d\zeta_2}{d\sigma} = -s\cos\sigma(\tilde{a}_{px}) + (\zeta_2 + (1+s)\sin\sigma)(\tilde{a}_{py})$$
$$\frac{d\zeta_3}{d\sigma} = -\zeta_3(\tilde{a}_{py})$$
TIMING
$$\frac{d\tau}{d\sigma} = \frac{1}{\zeta_3^3 s^2}$$
ADDITIONAL RELATIONS
$$s = 1 + \zeta_1\cos\sigma + \zeta_2\sin\sigma, \qquad \frac{1}{r} = \zeta_3^2 s$$
$$\frac{\mathrm{d}r}{\mathrm{d}\,\tau} = \zeta_3(\zeta_1\sin\sigma - \zeta_2\cos\sigma)$$
$$\tilde{a}_{px} = \frac{a_{px}}{\zeta_3^4 s^3} \qquad , \tilde{a}_{py} = \frac{a_{py}}{\zeta_3^4 s^3} \qquad \tilde{a}_{pz} = \frac{a_{pz}}{\zeta_3^4 s^3}$$

TABLE 3.2: Summary of equations for DROMO method

3.6 Comparing Formulations

Many similarities and differences can be pointed out between the Deprit and DROMO formulations.

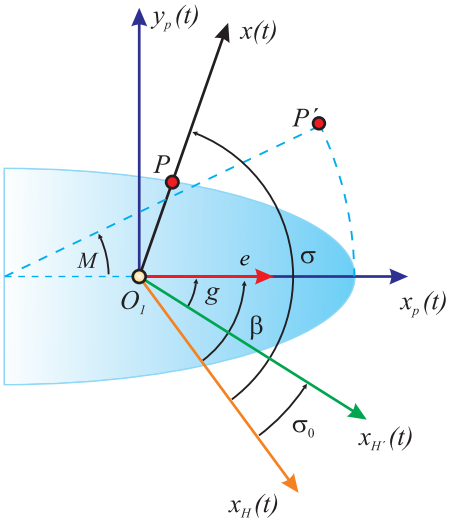


FIGURE 3.5: Geometrical relations between Deprit and DROMO

While both methods are built upon the ideal frame concept of Hansen (Ref.[22]), the ideal frames used in each formulation are different. Whereas for DROMO the ideal frame $\mathcal{H}$ is defined as coincident with the perifocal frame at the initial time, Deprit takes an ideal frame $\mathcal{H}'$ initially aligned with the orbital frame instead. Therefore, these ideal frames differ by a constant finite rotation of value σ_0 (see Fig.3.5), though both share the same fundamental properties.

In both methods, a set of Euler-Rodrigues parameters is used to describe the attitude of the corresponding ideal frame. Thus, quaternions for Deprit and for DROMO do not generally coincide, since the ideal frames they refer to are different, though they all vanish in the case of planar motions, as well as for the unperturbed case.

Both methods, use the angular momentum h as ideal element. Also, Deprit's elements (C, S) are equivalent to the DROMO elements (ζ_1, ζ_2), since they arise from the same concept: they represent the Cartesian projections of the eccentricity vector on the corresponding ideal frame. Note that, in Deprit, these values are also divided by the angular momentum h. So, these parameters fulfil the relation

$$\begin{bmatrix} C \\ S \end{bmatrix} = \frac{1}{h} \cdot \begin{bmatrix} \cos\sigma_0 & \sin\sigma_0 \\ -\sin\sigma_0 & \sin\sigma_0 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \end{bmatrix}$$

Finally, the major difference is encountered in the independent integration variable. The independent variable in DROMO is the ideal anomaly, σ, as the result of a Sundamnn transformation. Therefore, as the time, τ, does not appear explicitly, it has to be obtained along with the integration process as an additional dependent variable. The three main consequences are: 1) the formulation is unique for elliptic, parabolic and hyperbolic orbits, 2) no Kepler's Equation or equivalent must be solved, and 3) an integration event or stop condition is required to finalize the integration when $\tau = \tau_f$ is reached.

In Deprit's method, however, there is no such time regularization. So, the time, τ, is the independent variable, and the additional parameter F is introduced to locate the particle P along the orbit. The use of F imposes two main drawbacks as compared to regularized methods like DROMO: 1) as F is a sort of mean anomaly, the resulting formulation is valid just for elliptic orbits, and 2) a modified Kepler's Equation has to be solved in every integration step, which may handicap the performance.

3.7 Comparing Performances

Both methods have been implemented using FORTRAN 95 subroutines,and to the check and compare the performance the problem of Stiefel and Shiefel will be solved. It consists on a particle in an elliptical orbit with an inclination of 30^o and a great eccentricity $e = 0.95$. The unique perturbing forces of the problem are the i) Earth oblateness and ii) the Lunar perturbation. Besides, the COWELL method will be implemented given its simplicity and will be used to compare as well

Parameters associated to Earth gravitation are

$$J_2 = 1.08265 \times 10^{-3}, \qquad R_E = 6371.22Km, \qquad \mu = 398601.0Km^3s^{-2} \tag{3.59}$$

and Lunar perturbation is modelled with the force

$$\boldsymbol{F}_{PL} = -m\mu_L \left\{ \frac{\boldsymbol{r} - \boldsymbol{\rho}}{|\boldsymbol{r} - \boldsymbol{\rho}|^3} + \frac{\boldsymbol{\rho}}{\rho^3} \right\} \tag{3.60}$$

Where $\boldsymbol{r}$ and ρ are the position vectors of the satellite and Moon, respectively, in the inertial geocentrical frame and $\mu_L = 4902.66Km^3s^{-2}$. The Moon position is given by the following ephemeris

$$\boldsymbol{\rho} = \rho \left\{ \sin(\omega_L t), -\frac{\sqrt{3}}{2}\cos(\omega_L t), -\frac{1}{2}\cos(\omega_L t) \right\} \tag{3.61}$$

where $\rho = 384400Km$ and $\omega_L = 2.665315780887 \times 10^{-6}s^{-1}$.
The problem is determining the satellite's position after 50 revolutions(288.12768941 mean solar days), starting from the initial conditions

$$(r_1(t_0), r_2(t_0), r_3(t_0)) = (0.0, -5888.9727, -3400.0)Km \tag{3.62}$$

$$(\dot{v}_1(t_0), \dot{v}_2(t_0), \dot{v}_3(t_0)) = (10.691338, 0.0, 0.0)Kms^{-1} \tag{3.63}$$

which correspond to the perigee of the initial osculating orbit. The most precise calculus of final position $(r_1(t_f), r_2(t_f), r_3(t_f))$ is given in the reference

$$r_1(t_f) = -24219.0503Km \tag{3.64}$$

$$r_2(t_f) = 227962.1064Km \tag{3.65}$$

$$r_3(t_f) = 129753.4424Km \tag{3.66}$$

The problem is has been integrated using a Runge-Kutta-Fehlberg 7(8) algorithm of variable step size. All the computations have been done with the same computer (Intel Core i3) with the same compiler with Microsoft Visual Studio 2012.

To carry out comparisons in performance two factors have been selected, 1) Error: will be calculated as the quadratic error with respect to the Stiefel and Shiefel (S. and S.) solution, 2) Number of functions calls used to measure how costly the computation of the solution is.

In the following table we can compare results for a relative and absolute tolerance of the integrator of 10^{-15} for all the methods.

Method	**S. and S.**	**COWELL**	**DEPRIT**	**DROMO**
$x(Km)$	-24219.0503	-24219.05012	-24219.05035	-24219.049916
$y(Km)$	227962.1064	227962.106375	227962.106329	227962.106279
$z(Km)$	129753.4424	129753.442401	129753.442379	129753.442353
$Error(m)$		0.459869 E-2	0.9479 E-1	0.2254 E0
$Fcalls$		182081	57489	47024
$Runtime(s)$		0.457	0.405	0.161

TABLE 3.3: Comparisons of special perturbations methods

In this table we can observe the variety of results that each method provide. However, to carry a proper comparison a widely variety of tolerance must be choose, because each

method has it best performance at a different tolerance.

As we can see in Fig.(3.6), for a given precision DROMO requires the minimum function calls, with a performance that is superior than the others. Meanwhile, DEPRIT has an intermediate performance between DROMO and COWELL for error in the order of meters. However, when we intend to reduce this error, DEPRIT get worse. This could be provoked by solving Kepler's equation, which requires a high number of function calls to reach the needed precision. This is the biggest handicap for DEPRIT.

Despite having augmented the order of the system, DROMO obtain the best results with the less number of functions calls. This performance is due to its robustness and regularity.

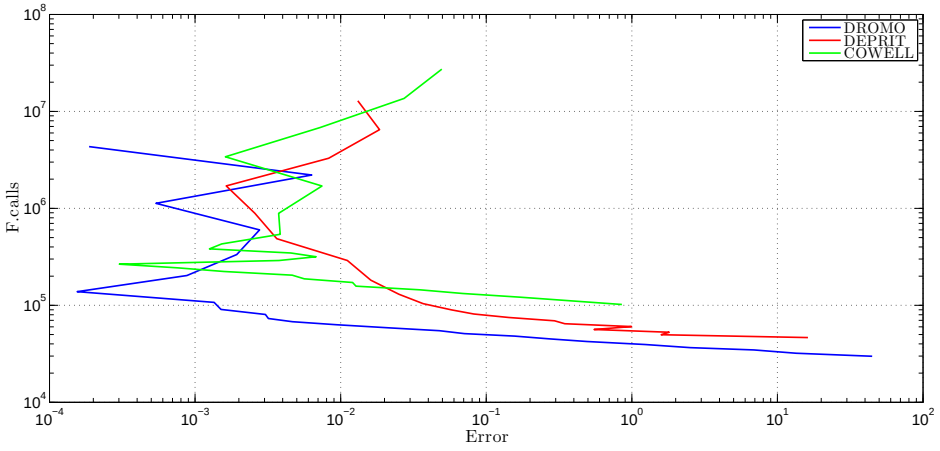


FIGURE 3.6: Function Calls v.s. Error for DROMO, Deprit and Cowell method

3.8 Conclusions

- DROMO and COWELL methods has a unique formulation for the three types of orbits: elliptic, parabolic and hyperbolic. However, DEPRIT is only useful for elliptical orbits, and singularities can be found in the proximity of parabolic motion, given that Kepler's equation cannot be solved, which is necessary to apply the method.
- Both DROMO and DEPRIT are based on the $ideal frames$ concept, and use the components of the eccentricity vector in this frame as variables, as well as the angular momentum.
- DROMO and DEPRIT used generalized variables (they take constant valued in the pure keplerian motion), and it means the truncation error in the non-perturbed

problem is eliminated and it is scaled by the perturbation itself in the perturbed one.

- Unlike Langrange's planetary equations, the three methods does not have singularities for small inclinations and/or small eccentricities. The attitude of the orbital plane in DEPRIT and DROMO is determined by the Euler parameters which are free of singularities.

- The use of Euler parameters also provides robustness and easy autocorrection, because when the module of the quaternion differs from 1, a proper normalization can be done.

- When compared with the COWELL method, both DEPRIT and DROMO requires and additional operation, since they use the components of perturbations forces in the orbital frame. However, this extra calculation,a dot product, does not jeopardize their performances as well .

- In DROMO and COWELL methods is not necessary to solve Kepler's equation in the elliptic case, since in DROMO time is one of the dependent variables determined by the method itself. However, in DEPRIT we have to solve it, it can caused a lack of precision and can be time-consuming.

- DROMO and DEPRIT integrate a system of eight differential equations to solve a sixth order problem. Although increasing the order of the system in two unities can be troublesome, there ir no disadvantages, as can be seen in the previous section.

Chapter 4

Brachistochrone Problem

4.1 Problem Statement

The brachistochrone problem was first posed by Johann Bernoulli in *Acta Eruditorum* in 1696 (can also be seen in Ref.[13]). The problem was the following:

"Given two points A and B in a vertical plane, what is the curve traced out by a point acted on only by gravity, which starts at A and reaches B in the shortest time..."

The problem can be mathematically formulated as follows. Let consider a particle situated at the origin O of a vertical plane OXY where Y is the vertical direction, directed downwards, and x is the perpendicular one. Let θ be the angle between the tangent to the trajectory and the vertical direction as we can see in Fig.(4.1).The only action acting on the particle is the gravitational acceleration, directed downwards.

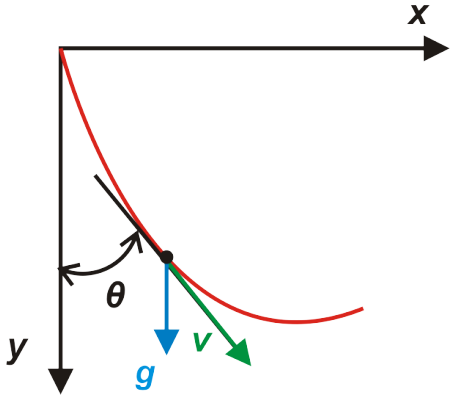


FIGURE 4.1: Brachistochrone problem

Projecting the velocity vector $\boldsymbol{v}$ in the plane OXY we can easily obtain

$$\dot{x} = v \sin\theta \tag{4.1}$$

$$\dot{y} = v \cos\theta \tag{4.2}$$

where $v = |\boldsymbol{v}|$.

Now, projecting the gravity vector along the velocity direction, we infer that

$$\dot{v} = g \cos\theta \tag{4.3}$$

This set of three differential equations represent the dynamical behaviour of the problem.

Next let us formulate the optimal control problem

$$\begin{aligned} &minimize && t_f && (4.4)\\ &subject \quad to && \dot{x} = v\sin\theta && (4.5)\\ & && \dot{y} = v\cos\theta && (4.6)\\ & && \dot{v} = g\cos\theta && (4.7) \end{aligned}$$

This problem has an analytical solution that can be obtained using the calculus of variations techniques. We will use this result to compare with the approximate solution obtained using optimal control methods.

4.2 Analytical Solution

Dividing the third and second equations in Eq.(4.4), we can easily obtain an expression for the velocity:

$$v = \sqrt{2gy} \tag{4.8}$$

But the velocity must also equal the rate of change of arc-length along the curve with respect to time, so we get

$$v = \frac{ds}{dt} = \frac{ds}{dx} \cdot \frac{dx}{dt} = \sqrt{1 + (y')^2}\frac{dx}{dt} \tag{4.9}$$

Equating the two formulas for velocity leads to

$$dt = \frac{\sqrt{1 + (y')^2}}{\sqrt{2gy}} dx \tag{4.10}$$

Which means that the total travel time for the particle is given by

$$t_f = \int_0^{x_f} \frac{\sqrt{1+(y')^2}}{\sqrt{2gy}} dx \tag{4.11}$$

So the mathematical statement of the problem is to find a continuously differentiable function, y, that minimizes t_f subject to $y(0) = 0$ and $y(t_f) = y_f$.

In order to solve the problem let consider the Euler-Lagrange equation

$$F_y - (\frac{d}{dx} F_{y'}) = 0 \tag{4.12}$$

As in this case, the optimal do not depend on x, the Eq.(4.12) can be reduced to

$$F - y' F_{y'} = C \tag{4.13}$$

where C is a constant. Substituting in this equation we get

$$\frac{\sqrt{1+(y')^2}}{y} - \frac{(y')^2}{\sqrt{y(1+(y')^2)}} = C \tag{4.14}$$

where the term $2g$ is included in the constant C. Algebraically simplifying

$$C^2(y)(1+(y')^2) = 1 \tag{4.15}$$

If we make the substitution $y = \frac{1}{C^2}(\sin(\beta/2))^2$, where β is a parameter, we get

$$y' = -\frac{\cos(\beta/2)}{\sin(\beta/2)} \tag{4.16}$$

where it is been chosen the negative slope to reflect the fact that initially the curve will have a negative slope.

From the chain rule $\frac{dy}{dx} = \frac{dy}{d\beta} \cdot \frac{d\beta}{dx}$, so

$$-\frac{cos(\beta/2)}{\sin(\beta/2)} = [\frac{1}{C^2}\sin(\beta/2)\cos(\beta/2)]\frac{d\beta}{dx} \longrightarrow dx = \frac{1}{C^2}\sin^2(\beta/2)d\beta \tag{4.17}$$

and we obtain a result for x, considering that $x = 0$ when $t = 0$.

$$x = \frac{1}{2C^2}(\beta - \sin\beta) \tag{4.18}$$

Let consider $A = \dfrac{1}{2C^2}$ son we can write the brachistochrone equations as follows

$$x = A(\beta - \sin\beta) \tag{4.19}$$

$$y = A(\cos\beta - 1) \tag{4.20}$$

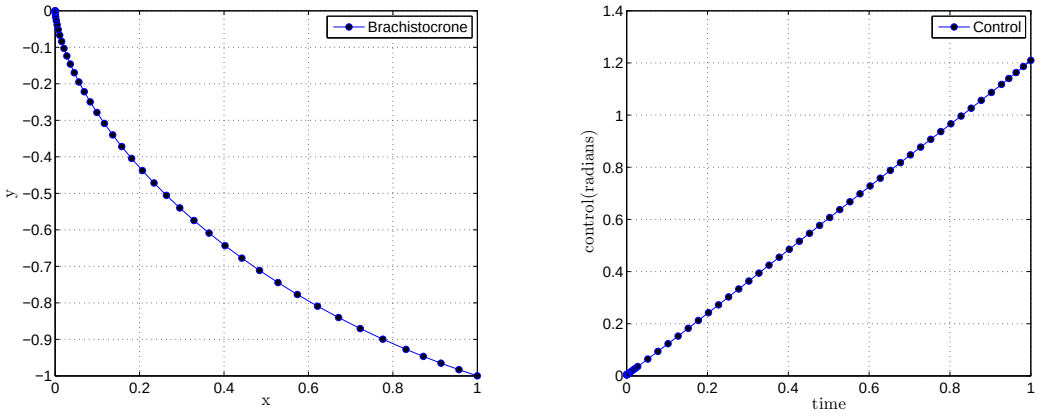
The constant A should be obtain satisfying the final conditions and t_f.

For the control θ we can obtain the analytical solution

$$\tan\theta = \frac{dx}{dy} = \frac{1-\cos\beta}{\sin\beta} \tag{4.21}$$

If we differentiate θ with respect to the parameter β , we can obtain a linear variation.

This analytical solution can be seen in Fig.(4.2), setting the final state to $[x_f, y_f] = [1, -1]$.



(A) Brachistochrone Trajectory (B) Brachistochrone Control

FIGURE 4.2: Brachistochrone solution

As we can observe, the control obtained is linear as the theory predicts.

4.3 Numerical Solution

In order to numerically solve the Optimal Control Problem (OCP), we implement two direct methods. This will enable to determine the efficient running of the software implemented. First we convert the OCP equations into a Non-linear problem formulation (NLP), and later we use state of art solvers.

Two different gradient-based solvers for large-scale nonlinear optimization will be used for the nonlinear optimization problem, namely, the Sequential Quadratic Programming student version for the solver SNOPT (Ref.[19]) (the number of variables is limited up to 300) and the Interior Point method IPOPT (Ref.[20]) (No limit on variables), both packages available in Matlab®. Gradients should be provided for IPOPT, and need not to be for SNOPT, because they can be calculated via Finite Differences (FD) by the program itself. A "function overloading" (free *myad* routine for Matlab) method for Automatic Differentiation is chosen to supply derivatives for the NLP solvers, and it will be compared with FD method of SNOPT. The Hessian will be approximated via a Limited-Memory BFGS (Broyden-Fletcher-Goldfarb-Sahnno) update method (Ref.[10]), used as default for both NLPs. Tolerances for both solvers will be set to 10^{-8}. This low tolerance was chosen to assure the convergence for all the numerical tests. In addition, variables are automatically scaled to maintain the gradient close to the unit.

Let us measure the performance for each solver by means of the quality of the solution in terms of accuracy, convergence and computational cost. The following outputs of the NLP:

- **Accuracy**: The accuracy or *Error* of the obtained solution is measured as the norm of the difference between the value of the cost functional obtained (t_f) and its value in a benchmark solution (t_f^*). The benchmark is set as the analytical solution. In addition, the parameter *Infeas.* evaluate the maximum infeasabilities in the constraints, that is, the maximum defect between the actual and the desired constraint value scaled with the norm of the vector of the NLP input variables. If this value is lower than 10^{-8} the problem is considered infeasible.

$$Error = |t_f - t_f^*|/|t_f^*| \tag{4.22}$$

- **Convergence** : We consider the total amount of iterations (*N. Iter.*) of the Newton-based method, that is, the number of times the corresponding KKT system is solved to reach the solution, as a measure of the convergence for each method. A lower number of iterations means better convergence properties.

- **Computational cost**: The computational cost will be measured in two ways. On the one hand, the number of function calls *F.calls*, that is, the times the function for the constraints is called. On the other hand, the computational *Runtime*, in seconds, measures the time needed for the solver to optimize the problem.

Le us check the direct shooting method, which consists on parametrizing the control by a finite set of elements (7 in this case):

$$\theta = p_1 + p_2 t + p_3 t^2 + p_4 t^3 + p_5 t^4 + p_6 t^5 + p_7 t^6 \tag{4.23}$$

Then we integrate the ODE system with and initial value method and obtain the final state. As merit function we chose the difference between the desired and the obtained final state. Thus the variables for SNOPT to be optimized are the control finite set of parameters, in order to fit the required final state.

Moreover, my implementation of the collocation method approximating the state time history by a cubic Hermite interpolant as has been explained in (Sec.2.3.2.2) will be tested. The control will be linearly approximated.

To carry on the simulation the final state will be fixed as $[x_f, y_f] = [1, 1]$.

Let consider the final time t_f in order to normalize time, thus the time interval of integration convert from $[0, t_f]$ to $[0, 1]$.

$$\frac{d}{dt} = \frac{d}{t_f d\tau} \tag{4.24}$$

Hence, the final time is treated as an additional variable for the program.

As SNOPT and IPOPT are iterative methods, they need and initial guess. In order to start running all the solvers from the same situation, the same initial guess will be used. The control is set continuously equal to 1 and the variables x and y are linearly approximated between the initial and final state. The initial guess for the final time is chosen as the unit.

4.4 Comparisons

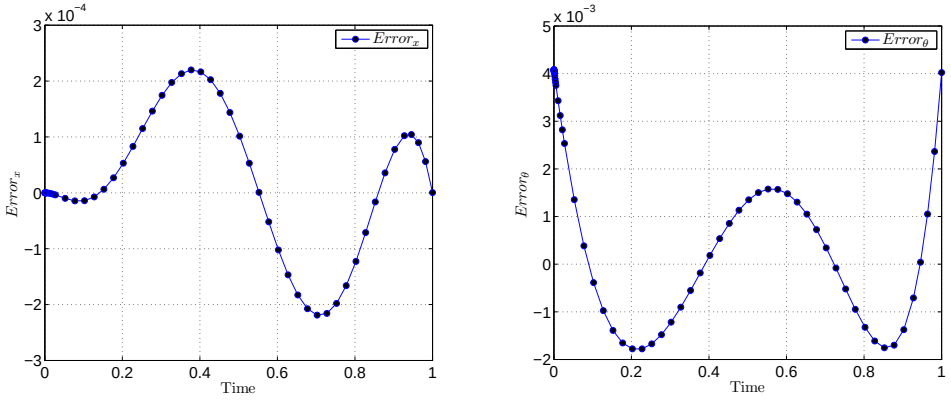
In Tab.(4.1) the numerical performances of the solvers for $N = 20$ nodes are shown. Single shooting is the easiest to implement, and has moderate performances. However, it is not possible to achieve the same accuracy as with collocation methods. For this number of nodes, using AD instead of FD produces the same accuracy, with more computational cost, because although FD requires 1000 more function calls to compute gradients, this function calls are faster than those in AD where the value and the Jacobian of the constraints are computed at the same time, involving additional operations. The Interior Point solver IPOPT performs the best, obtaining a solution one order more precise in only $0,2$ seconds.

In Figs.(4.3)-(4.6) the difference between the analytical solution and the one numerically obtained is plotted at all the nodes considered.

The collocation and single shooting implementations have been successfully tested performing well, approximating the solution as accurate as 10^{-8}. Performances of this methods and for the NLP solvers, will be analysed in detail in Chap.(5) and Chap.(6)

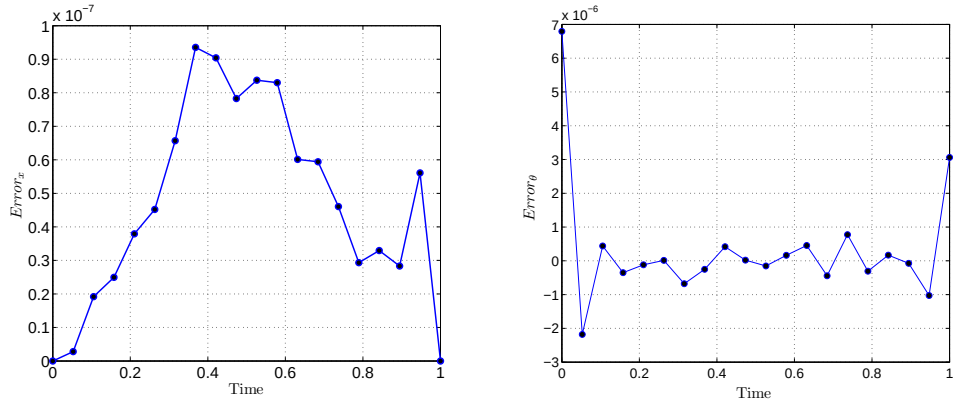
TABLE 4.1: SNOPT and IPOPT performance for Single Shooting and Direct Collocation method

Method	Solver	Infeas.	Iter.	F.calls	Error	Runtime(s)
SS	SNOPT(FD)	8,60E-12	41	394	1E-04	1,32
DC	SNOPT(FD)	2,70E-13	69	1912	1E-07	0,54
DC	SNOPT(AD)	5,00E-13	69	68	1E-07	1,66
DC	IPOPT(AD)	7,13E-09	22	23	1E-08	0,20



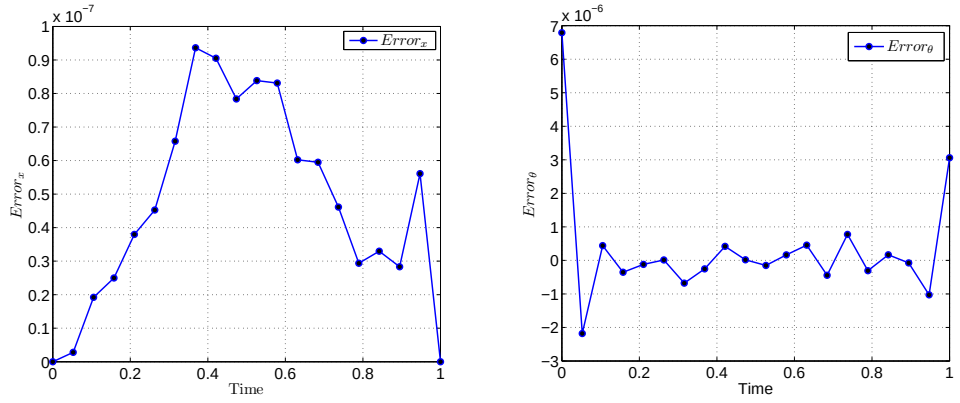
(A) Error in x (B) Error in θ

FIGURE 4.3: SNOPT Error using Single Shooting method and FD



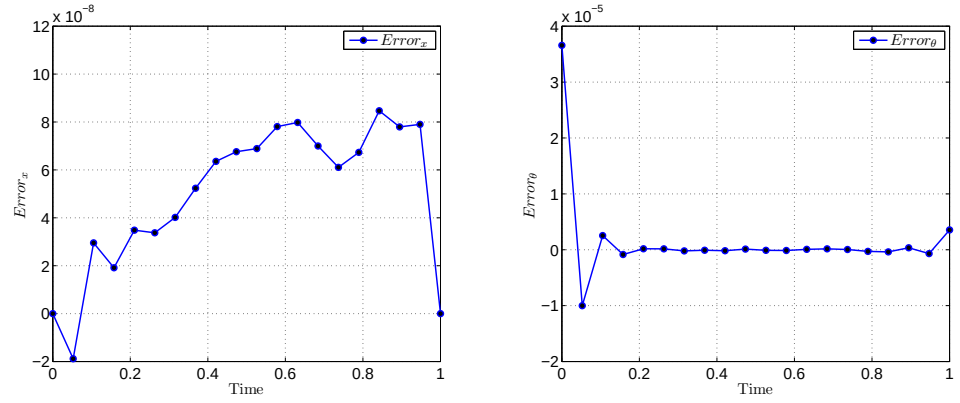
(A) Error in x (B) Error in θ

FIGURE 4.4: SNOPT Error using Direct Collocation method and FD



(A) Error in x (B) Error in θ

FIGURE 4.5: SNOPT Error using Direct Collocation method and AD



(A) Error in x (B) Error in θ

FIGURE 4.6: IPOPT Error using Direct Collocation method and AD

Chapter 5

Maximum Radius Orbit Transfer

5.1 Problem Statement

The classical Optimal Control Problem "*Maximum radius orbit transfer in a given time*" was described on pages 66-69 of the classic text Applied Optimal Control, by Arthur E. Bryson, Jr. and Yu-Chi Ho in Ref.[13]. The problem was stated as follows:

"Given a constant-thrust rocket engine, T=thrust, operating for a given length of time, t_f, we wish to find the thrust-direction history, $\phi(t)$, to transfer a rocket vehicle from a given initial circular orbit to the largest possible circular orbit."

The nomenclature is defined in the figure 5.1.

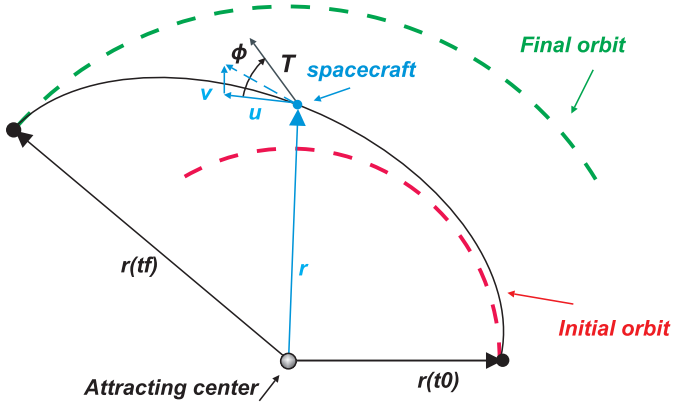


FIGURE 5.1: Maximum Radius problem

where

r = radial distance of spacecraft from attracting center
u = radial component of velocity
v = tangential component of velocity
m_0 = initial mass of the spacecraft
$\dot{m}$ = fuel consumption rate (constant)
ϕ = thrust angle
μ = gravitational constant of attracting center

The numerical example given by Arthur E.Bryson (Ref.[13]) is a coplanar orbit from Earth to Mars: The orbit of each planet is assumed circular and the transfer time is about 193 days. The initial mass and propulsive characteristics are as follows

$$T = 3.781N \tag{5.1}$$

$$m_0 = 4535.9kg \tag{5.2}$$

$$\dot{m} = 5.85kg/day \tag{5.3}$$

To construct a proper normalization, μ/r_0^2 will be used as the acceleration unit, and r_0^3/μ as time unit,where r_0 is the radius of the Earth's orbit and μ is the Sun's gravitational constant. Hence, the non-dimensional acceleration due to thrusting is given by

$$\frac{T/m_0}{\mu/r_0^2} = 0.1405 \tag{5.4}$$

The non-dimensional total time flight is given by

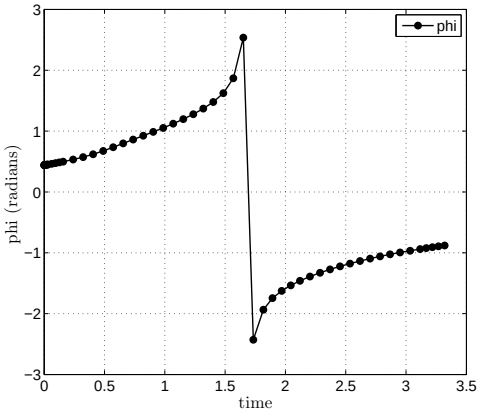
$$\frac{t_f}{\sqrt{r_0^3/\mu}} = 3.32 \tag{5.5}$$

and the non-dimensional flow rate

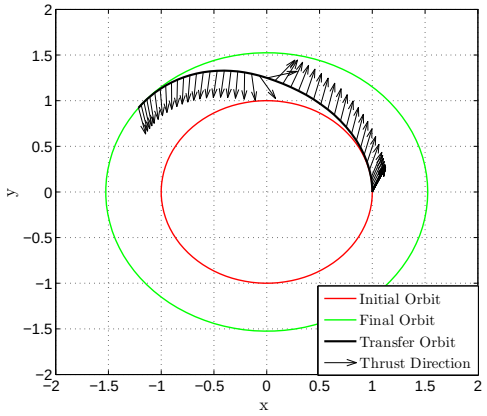
$$|\dot{m}|\sqrt{\mu/r_0}/T = 0.5328825 \tag{5.6}$$

In Fig.(5.2b) the trajectory (black) is plotted in the orbital plane delimited by the initial circular orbit (red) and the final circular orbit (green), where the center correspond to the main attracting body. In Fig.(5.2a) we can observe the solution for the control as an antisymmetric function with respect to $tf/2$. The control slowly increases up to $\pi/2 \quad rad$ and continues with a rapid continuous variation up to $-\pi/2 \quad rad$, and finally decreases. In conclusion, the thrust is directed inwards the orbit the first half of the

trajectory and outwards the remaining half. This solution has been obtained via indirect collocation method using the Matlab function **bvp4c** with polar variables.



(A) Control solution



(B) Trajectory solution

5.1.1 Polar formulation

This Optimal Control Problem can be formulated in terms of the set of Polar variables (r, u, v, t), where t represents the dimensional time, as

Find the control time-history $\phi(t)$ *that maximizes* $r_f = r(t_f)$.

The dynamics is governed by the first-order, two-dimensional equations of motion

$$\frac{dr}{dt} = u \tag{5.7}$$

$$\frac{du}{dt} = \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T \sin\phi}{m_0 - |\dot{m}|t} \tag{5.8}$$

$$\frac{dv}{dt} = -\frac{uv}{r} + \frac{T \cos\phi}{m_0 - |\dot{m}|t} \tag{5.9}$$

and must be solved given the *initial conditions*

$$r(t_0) = r_0, \qquad u(t_0) = 0, \qquad v(t_0) = \sqrt{\mu/r(t_0)} \tag{5.10}$$

and subject to the *terminal constraints*

$$u(t_f) = 0, \qquad v(t_f) = \sqrt{\mu/r(t_f)} \tag{5.11}$$

Using the Calculus of Variations (Sec.2.2.2), the *Hamiltonian* has the following expression:

$$\mathcal{H} = \lambda_r u + \lambda_u(\frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T\sin\phi}{m_0 - |\dot{m}|t}) + \lambda_v(-\frac{uv}{r} + \frac{T\cos\phi}{m_0 - |\dot{m}|t}) \tag{5.12}$$

and

$$\Phi = r(t_f) + \nu_1 u(t_f) + \nu_2[v(t_f) - \sqrt{\frac{\mu}{r(t_f)}}] \tag{5.13}$$

Thus, the equations for the costates become

$$\frac{d\lambda_r}{dt} = -\lambda_u(-\frac{v^2}{r^2} + \frac{2\mu}{r^3}) - \lambda_v(\frac{uv}{r^2}) \tag{5.14}$$

$$\frac{d\lambda_u}{dt} = -\lambda_r + \lambda_v\frac{v}{r} \tag{5.15}$$

$$\frac{d\lambda_v}{dt} = -\lambda_u\frac{2v}{r} + \lambda_v\frac{u}{r} \tag{5.16}$$

and they are subject to the terminal conditions

$$\lambda_r(t_f) = 1 + \frac{\nu_2\sqrt{\mu}}{2[r(t_f)]^{3/2}}, \qquad \lambda_u(t_f) = \nu_1, \qquad \lambda_v(t_f) = \nu_2 \tag{5.17}$$

Finally, the control can be obtained by using the Pontryagin's maximum principle as

$$0 = (\lambda_u\cos\phi - \lambda_v\sin\phi)\frac{T}{m_0 - |\dot{m}|t} \longrightarrow \tan\phi = \frac{\lambda_u}{\lambda_v} \tag{5.18}$$

Hence, the control can be determined in terms of λ_u and λ_v. Additionally, ν_1 and ν_2 should be chosen to satisfy the terminal conditions (Eq.(6.6)).

5.1.2 DROMO formulation

The Trajectory Optimization Problem can be reformulated using DROMO variables in the particular planar case $[\zeta_1, \zeta_2, \zeta_3, \tau, \sigma]$, whose formulation is described in 3.5. Let us remark that the DROMO independent variable is the angular parameter σ instead of time τ. The following notation is introduced in order to simplify the formulation:

$$s = 1 + \zeta_1\cos\sigma + \zeta_2\sin\sigma, \qquad p = \zeta_1 + (1+s)\cos\sigma, \qquad q = \zeta_2 + (1+s)\sin\sigma \tag{5.19}$$

$$\tilde{a}_{px} = \frac{T\sin\phi}{m_0 - |\dot{m}|\tau}/(\zeta_3^4 s^3), \qquad \tilde{a}_{py} = \frac{T\cos\phi}{m_0 - |\dot{m}|\tau}/(\zeta_3^4 s^3) \tag{5.20}$$

The optimal control problem can be state as follows:

$$\textit{Find } \phi(\tau) \textit{ that maximizes } r(\tau_f) = 1/(\zeta_3^2(\sigma_f)s_f)$$

The system is governed by the following differential equations:

$$\frac{d\zeta_1}{d\sigma} = s\sin\sigma(\tilde{a}_{px}) + p(\tilde{a}_{py}) \tag{5.21}$$
$$\frac{d\zeta_2}{d\sigma} = -s\cos\sigma(\tilde{a}_{px}) + q(\tilde{a}_{py}) \tag{5.22}$$
$$\frac{d\zeta_3}{d\sigma} = -\zeta_3(\tilde{a}_{py}) \tag{5.23}$$
$$\frac{dt}{d\sigma} = \frac{1}{\zeta_3^3 s^2} \tag{5.24}$$

and must be solved given the *initial conditions*

$$\zeta_1(\sigma_0) = 0, \qquad \zeta_2(\sigma_0) = 0, \qquad \zeta_3(\sigma_0) = 1/\sqrt{\mu r_0}, \qquad \tau(\sigma_0) = \tau_0 \tag{5.25}$$

and subject to the *terminal constraints*

$$\zeta_1(\sigma_f) = 0, \qquad \zeta_2(\sigma_f) = 0, \qquad \tau(\sigma_f) = \tau_f \tag{5.26}$$

The corresponding *Hamiltonian* can be formulated as

$$\mathcal{H} = \lambda_{\zeta_1}(s\sin\sigma(\tilde{a}_{px}) + p(\tilde{a}_{py})) + \lambda_{\zeta_2}(-s\cos\sigma(\tilde{a}_{px}) + q(\tilde{a}_{py})) \tag{5.27}$$
$$+\lambda_{\zeta_3}(-\zeta_3(\tilde{a}_{py})) + \lambda_\tau(1/(\zeta_3^3 s^2)) \tag{5.28}$$

and

$$\Phi = \frac{1}{\zeta_3(\sigma_f)^2} + \nu_1(\zeta_1(\sigma_f)) + \nu_2(\zeta_2(\sigma_f)) + \nu_3(\tau(\sigma_f) - \tau_f) \tag{5.29}$$

Thus, the equation of the costates become

$$\frac{d\lambda_{\zeta_1}}{d\sigma} = -\lambda_{\zeta_1}[-\sin(2\sigma)\tilde{a}_{px} + (1+\cos^2(\sigma) - \frac{3}{s}p\cos\sigma)\tilde{a}_{py}] - \lambda_{\zeta_2}[(2\cos^2\sigma)\tilde{a}_{px} + (\sin\sigma\cos\sigma - \frac{3}{s}q\cos\sigma)\tilde{a}_{py}] - \lambda_{\zeta_3}[(\frac{3}{s}\zeta_3\cos\sigma)\tilde{a}_{py}] + \lambda_\tau[\frac{2}{\zeta_3^3 s^3}\cos\sigma] \tag{5.30}$$

$$\frac{d\lambda_{\zeta_2}}{d\sigma} = -\lambda_{\zeta_1}[-2\sin^2(\sigma)\tilde{a}_{px} + (\sin\sigma\cos\sigma - \frac{3}{s}p\sin\sigma)\tilde{a}_{py}] - \lambda_{\zeta_2}[(\sin(2\sigma))\tilde{a}_{px} + (1+\sin^2\sigma - \frac{3}{s}q\sin\sigma)\tilde{a}_{py}] - \lambda_{\zeta_3}[(\frac{3}{s}\zeta_3\sin\sigma)\tilde{a}_{py}] + \lambda_\tau[\frac{2}{\zeta_3^3 s^3}\sin\sigma] \tag{5.31}$$

$$\frac{d\lambda_{\zeta_3}}{d\sigma} = +\lambda_{\zeta_1}[s\sin\sigma(\tilde{a}_{px}) + p(\tilde{a}_{py})](\frac{4}{\zeta_3}) + \lambda_{\zeta_2}[-s\cos\sigma(\tilde{a}_{px}) + q(\tilde{a}_{py})](\frac{4}{\zeta_3}) - \lambda_{\zeta_3}[3\tilde{a}_{py}] + \lambda_\tau[\frac{3}{\zeta_3^4 s^2}] \tag{5.32}$$

$$\frac{d\lambda_\tau}{d\sigma} = -\lambda_{\zeta_1}[s\sin\sigma(\tilde{a}_{px}) + p(\tilde{a}_{py})]\frac{|\dot{m}|}{m_0 - |\dot{m}|\tau} - \lambda_{\zeta_2}[-s\cos\sigma(\tilde{a}_{px}) + q(\tilde{a}_{py})]\frac{|\dot{m}|}{m_0 - |\dot{m}|\tau} + \lambda_{\zeta_3}[\zeta_3\tilde{a}_{py}]\frac{|\dot{m}|}{m_0 - |\dot{m}|\tau} \tag{5.33}$$

The system is completed with the following *terminal adjoint conditions*:

$$\lambda_{\zeta_1}(\sigma_f) = \nu_1, \qquad \lambda_{\zeta_2}(\sigma_f) = \nu_2, \qquad \lambda_{\zeta_3}(\sigma_f) = -\frac{2}{\zeta_3(\sigma_f)^3}, \qquad \lambda_\tau(\sigma_f) = \nu_3 \tag{5.34}$$

Finally, the control can be determined as a function of the costates using the Pontryagin's maximum principle as:

$$\lambda_{\zeta_1}(s\sin\sigma\cos\phi - p\sin\phi) + \lambda_{\zeta_2}(-s\cos\sigma\cos\phi - q\sin\phi) + \lambda_{\zeta_3}(\zeta_3\sin\phi) = 0 \tag{5.35}$$

$$\tan\phi = \frac{\lambda_{\zeta_1} s\sin\sigma - \lambda_{\zeta_2} s\cos\sigma}{\lambda_{\zeta_1} p + \lambda_{\zeta_2} q - \lambda_{\zeta_3}\zeta_3} \tag{5.36}$$

Note that, for DROMO formulation, the final state of the independent variable is unknown. Hence, it needs to be considered as an additional parameter which should be calculated during the optimization process.

5.2 Numerical Solutions

We are intended to compare performances between the three methods implemented to solve OCP, namely, single shooting, collocation and pseudospectral, using different NLP solvers for both indirect and direct approaches. In addition, the effect of using a high precision orbit propagator DROMO, instead of the classical Polar formulation, as it has

been proven to provide accurate solutions in a lower time among others propagators (Ref.[11]), will be examined. We transcribe the OCP to the Nonlinear Optimization solvers SNOPT and IPOPT. Different ways to provide gradients will be considered in order to select the most appropriate: Finite Differences (FD), Complex Differentiation (CD) and Automatic Differentiation, both "function overloading" (AD) and "source transformation" (AD*).

For the sake of clarity, all the numerical tests carried out are summarized in Tab.(5.1)

TABLE 5.1: Numerical tests summary

Method	Problem	Variables	SNOPT			IPOPT	
			FD	AD	CD	AD	AD*
S.SHOOTING	Direct	Polar	x				
		Dromo	x				
	Indirect	Polar	x				
		Dromo	x				
COLLOCATION	Direct	Polar	x	x		x	x
		Dromo	x	x		x	
	Indirect	Polar	x	x		x	
		Dromo	x	x		x	
PSEUDOSPECTRAL	Direct	Polar	x	x	x		
		Dromo		x			

The way to compare performances has been established in Sec.(4.3). The Benchmark solution for this problem is set as the solution for the indirect problem obtained via the Matlab® function *bvp4c* using Polar variables. Tolerances for both solvers will be set to 10^{-6}. This low tolerance was chosen to assure the convergence for all the numerical tests. In addition, variables are automatically scaled to maintain the gradient close to the unit.

As iterative Newton-based solvers are used, it is necessary to set the same initial guess for all the numerical simulations. It is desirable,that the initial guess would fit the boundary conditions. To achieve this objective, a simpler problem will be assumed. Let us consider the unperturbed problem case, were the thrust magnitude is zero ($T = 0$). As the time of integration is quite small, the spacecraft not even complete a whole orbit around the primary, so that the dynamics is expected to not widely vary. This assumption is not normal for low-trust transfer where the time of integration are quite large. Setting this condition, an easy solution for both Hamiltonian system (Eq.(6.7)) and (Eq.(6.26)) can be found. The solution correspond to a constant set of states and costates for both Polar and DROMO variables.

In the Polar problem the solution can be expressed as follows:

$$r(t) = 1, \quad u(t) = 0, \quad v(t) = 1, \quad \lambda_u(t) = 0, \quad \lambda_v(t) = 2; \quad \lambda_r(t) = 2 \tag{5.37}$$

Assuming the same no-thrusting conditions for DROMO variables, we infer the following solution:

$$\begin{aligned} \zeta_1(\sigma) = 0, \quad \zeta_2(\sigma) = 0, \quad , \zeta_3(\sigma) = 1, \quad \tau(\sigma) = \tau_0 + \sigma, \\ \lambda_{\zeta_1} = 0, \quad \lambda_{\zeta_2} = 0, \quad \lambda_{\zeta_3} = -2, \quad \lambda_\tau = 0 \end{aligned} \tag{5.38}$$

Enforcing that $\tau(\sigma_f) = 3.32$ we obtain an initial guess for the final value of the independent variable $\sigma_f = 3.32$.

Finally, as initial guess, the control is considered continuously tangent to the orbit, directed towards the velocity vector.

This initial guesses are appropriate as they allow us to solve the problem using all the methods proposed.

5.3 Numerical Solution by Single shooting

Let us start the comparisons applying the single shooting method, which is the simplest to implement. Both direct and indirect shooting methods allow us to integrate the dynamics taking advantages of the time-marching methods as explained in Sec.(2.3.2.1). The key topic in this methods consists on choosing the appropriate ODE solver. Given that we want to maintained a fixed tolerance to solve the dynamics independently of the method used, direct or indirect, variable-step size will be chosen. Among this solvers we will focus on three different ones which provide the following performances :

- **ode45** : It is an explicit one-step method. In general, this Runge-Kutta(4,5) solver is the best to apply as a first try for most problems. For this reason, ode45 is the default solver for models with continuous states.
- **ode113**: It is an explicit multistep-method Adams-Bashforth-Moutlon. For problems with stringent error tolerances or for computationally intensive problems, this solver can be more efficient than ode45.
- **ode23s**: It is an implicit one-step method Second-order, modified Rosenbrock formula. It can be the best option to solve stiff problems, where the other two

solvers are useless. A stiff system is a system that has extremely different time scales

However, this routines cannot deal with AD-objects, so in this case, only the SNOPT solver can be used providing gradients via Finite Differences (FD).

5.3.1 Direct shooting

This method boasts of flexibility, where several approximations of the control can be considered. Here, a global polynomial approximation and a piece-wise lineal approximation will be used.

5.3.1.1 Polynomial approximation

In this case, given that the control is expressed as the arctan of a rational function as we deduce from both POLAR (Eq.(6.1.1)) and DROMO (Eq.(6.1.2)) formulations, a 4-th order polynomial approximation in the interval $[t_0, t_f]$ of the numerator and the denominator is proposed, where t represent the independent variable (τ for POLAR and σ for DROMO).

$$\phi = \arctan \frac{p_{11} + p_{12}t + p_{13}t^2 + p_{14}t^3 + p_{15}t^4}{p_{21} + p_{22}t + p_{23}t^2 + p_{24}t^3 + p_{25}t^4} \tag{5.39}$$

In Table (5.2) a summary of the performance for SNOPT using different integrators is shown, respectively with Polar and DROMO variables. It can be seen that using only a small number of optimization parameters (8 for Polar and 9 for DROMO), quite accurate results can rapidly be obtained. For this problem, only the Runge-Kutta algorithm ode45 converges to the solution with a quite accurate result. The implicit scheme ode23s is more time-consuming as it needs an extra computation at each step (to solve the implicit state equation by a Newton-based method (Ref.[17])), and is not appropriate for this problem because is non-stiff.

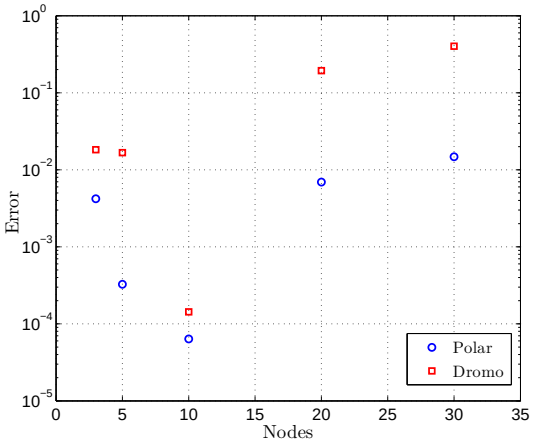
The use of DROMO variables improve the accuracy while reducing the number of Iterations and functions calls. However, it needs more time to be solved, because DROMO formulation, having one more variable than Polar, takes more time to compute gradients.

TABLE 5.2: SNOPT performance for Direct Single Shooting Method using a polynomial approximation of the control

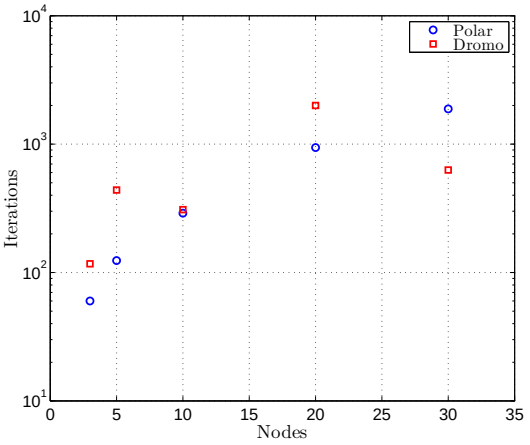
ODE solver	Variables	Infeas.	Iter.	F.calls	Error	Runtime(s)
ODE45	Polar	3,03E-07	125	1525	3,03E-05	8,48
ODE45	Dromo	1,20E-08	59	641	2,56E-06	10,02
ODE23S	Polar	7,80E-03	157	1938	4,62E-02	35,55
ODE23S	Dromo	3,60E-05	188	1888	3,45E-03	68,63
ODE113	Polar	2,70E-03	62	593	1,35E-01	6,18
ODE113	Dromo	4,30E-01	63	484	7,18E-01	11,03

5.3.1.2 Piecewise linear approximation

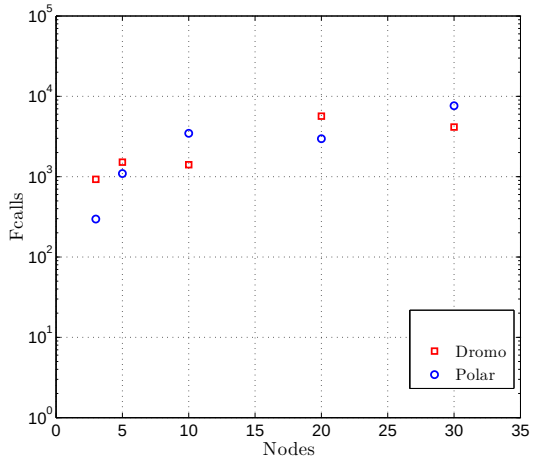
Let us now propose a piecewise linear approximation of the control (Eq.(2.21)) in a uniform grid along the interval $[t_0, t_f]$, where t represent the independent variable (τ for POLAR and σ for DROMO). In Tab.(5.4) the performance of SNOPT is shown as a function of the number N of nodes, respectively for Polar and DROMO variables. As we can infer from both tables, increasing the number of nodes worsen the convergence (Fig.(5.3)), when $N > 10$ the Error tends to increase. This is related with the use of Finite Differences, whose limitations (Sec.2.3.4) can lead to a bad-convergence or even to non-convergence.



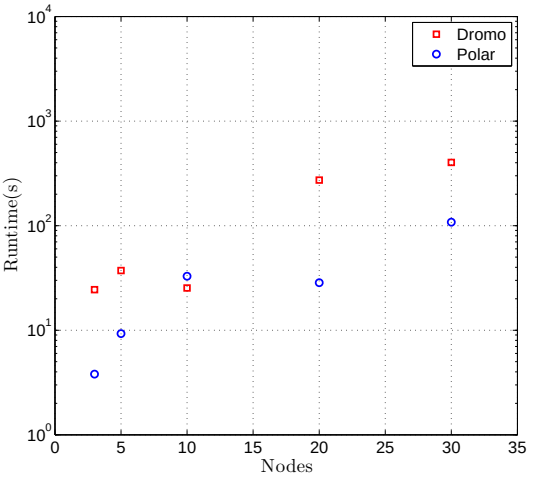
(A) Error vs N.nodes



(B) Iterations vs N.nodes



(C) F.calls vs N.nodes



(D) Runtime vs N.nodes

FIGURE 5.3: SNOPT performance for Direct Single shooting method using piecewise-linear approximation for the control for Polar and Dromo variables

5.3.2 Indirect shooting

Performances of SNOPT solving the states and costates dynamical equations with a Single shooting method, applying different ODE solvers, are summarized in Table (5.3). In this case, the convergence is fast accomplished with the Runge-Kutta integrator ode45, with less computational effort than in the direct case. Only 2 iterations are needed, compared to hundreds of iterations for the Direct case with Polar variables, even using only 6 variables. The complex indirect DROMO system, cannot be solved via this method, due to the errors in calculating gradients introduced by the Finite Differences method.

TABLE 5.3: SNOPT performance for linear approximation of the control of Indirect Single Shooting Method

Method	ODE solver	Variables	Infeas.	Iter.	F.calls	Error	Runtime(s)
SS	ODE45	Polar	1,50E-07	2	82	8,59E-05	1,10
SS	ODE45	Dromo	5,80E-01	9	140	1,53E-02	2,07
SS	ODE23S	Polar	3,50E-10	2	113	1,45E-04	3,26
SS	ODE23S	Dromo	1,60E+00	12	155	5,46E-02	7,47
SS	ODE113	Polar	3,47E-01	24	185	3,47E-01	1,94
SS	ODE113	Dromo	2,00E+01	32	142	5,94E-02	3,24

5.4 Numerical Solution by Collocation

In this section, the most common method for solving a OCP (Ref.[5]), known as Collocation method will be used, approximating the state/costate time history by a cubic Hermite interpolant as has been explained in (Sec.2.3.2.2), and the control will be linearly approximated. Moreover the need to introduce Automatic Differentiation techniques to compute gradients will be justified.

5.4.1 Direct Collocation

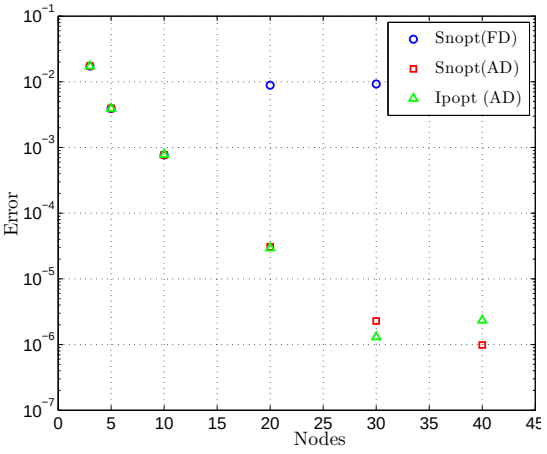
In Fig.(5.4) and Fig.(5.5) it is shown the performance of the method as a function of the number of Nodes for Polar and DROMO formulations respectively. Corresponding data are summarizes in Table (5.5). As the number of nodes increases the computational time needed to evaluate the Finite Differences for SNOPT become too large and lead to non-convergence of the method, for both Polar and DROMO. Note the big amount of computational effort done by Finite Differences in DROMO formulation, due to the complexity of the DROMO system. No more data for $N > 30$ for Polar , and $N > 10$ for Dromo were carried out because the simulation was too much time consuming.

As a solution, Automatic Differentiation is proposed. Both, "function overloading" (AD) and "source transformation"(AD*) will be used. By using AD and AD*, derivatives are computed up to machine precision, as if they were analytically computed, so that the number function calls of the solver is reduced because in the same call they provide the corresponding value and gradient. With a reduced number of Nodes $N < 10$ there is no much different between FD, but exceeding this number, AD and AD* reduces one order the iterations and Runtime. AD is more efficient than AD* as they involves less computational time. In AD the chain rule is applied directly by performing the same

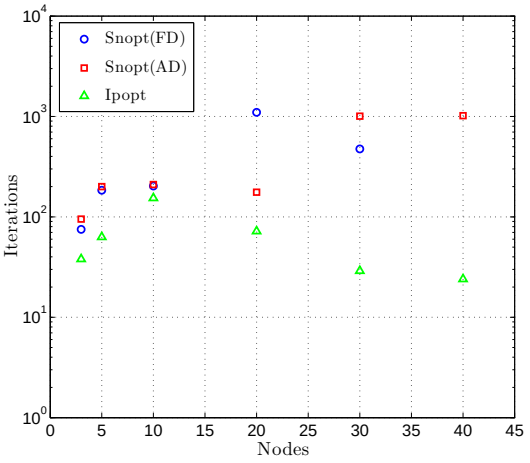
number of operations as the dynamical system itself treating the NLP variables as special objects. It limits the flexibility of this tools as not all the operations can be performed. However, for AD*, a new code is generated were more operations are involved, although is more flexible.

In addition, the IP solver IPOPT is seen to perform better as the number of nodes increases, reducing the number of iterations, runtime and F.calls to get a more accurate solution. Even, there exits a maximum for the iterations and Runtime, due to the fact, that is easier for the solver to adapt the approximated solution as the number of nodes increases.

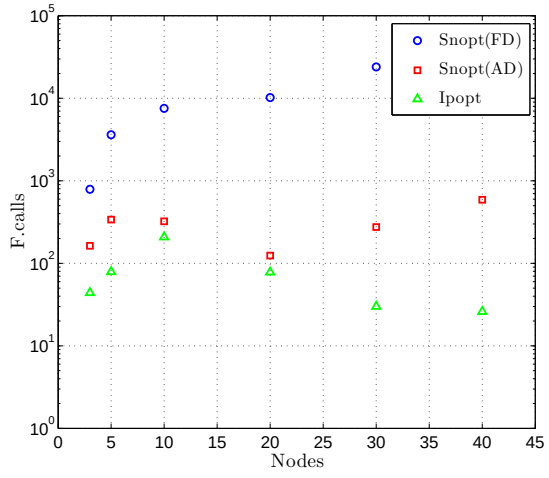
In general, SNOPT need more iterations to converge than IPOPT, since it is expected when comparing an active-set method to an interior point one. Hence, IPOPT appears to be more robust than SNOPT.



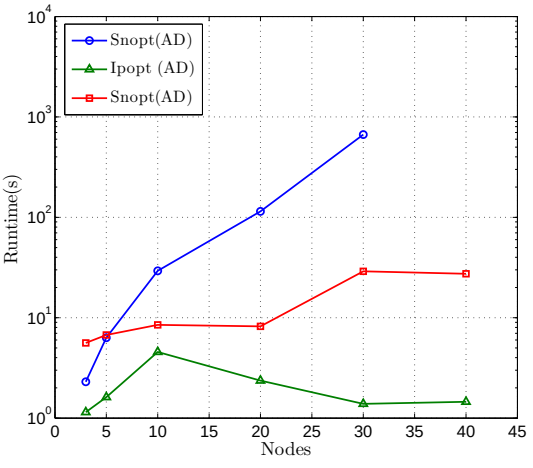
(A) Error vs N.nodes



(B) Iterations vs N.nodes

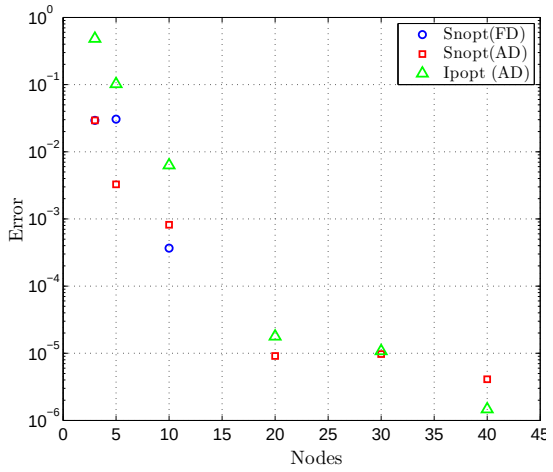


(C) F.calls vs N.nodes

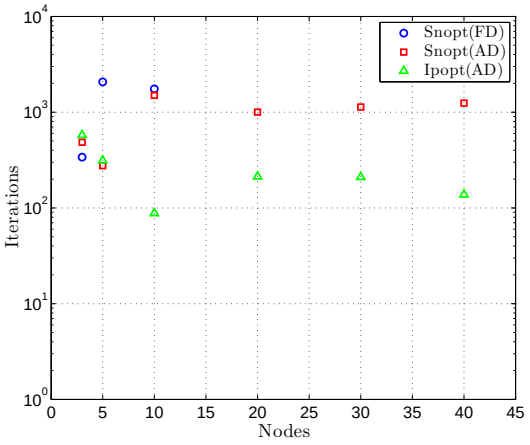


(D) Runtime vs N.nodes

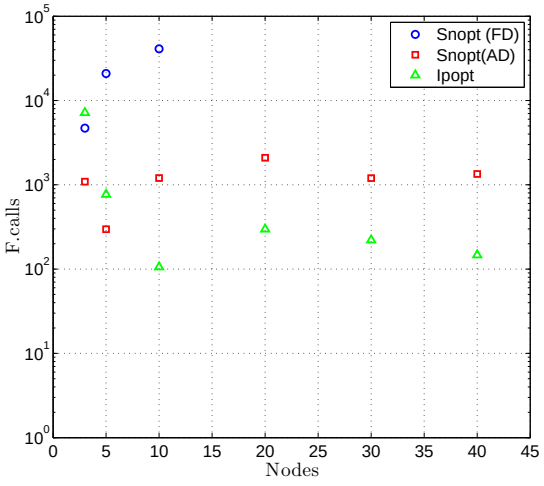
FIGURE 5.4: SNOPT and IPOPT performance for Direct Collocation method using Polar variables



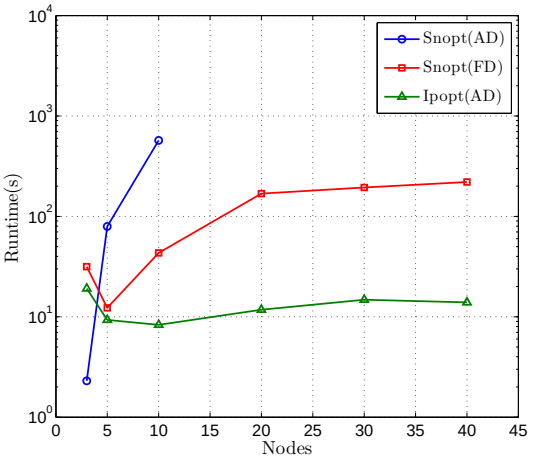
(A) Error vs N.nodes



(B) Iterations vs N.nodes



(C) F.calls vs N.nodes

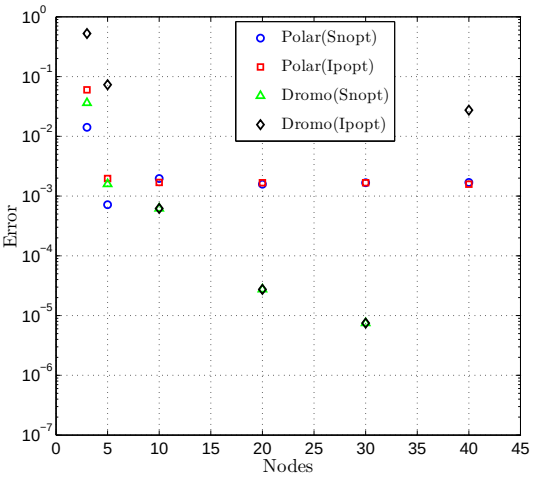


(D) Runtime vs N.nodes

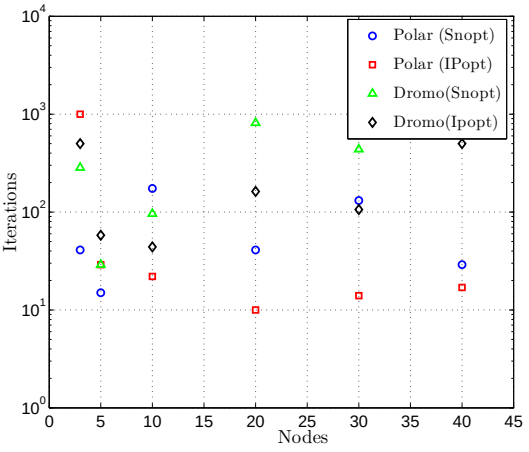
FIGURE 5.5: SNOPT and IPOPT performance for Direct Collocation method using Dromo variables

5.4.2 Indirect Collocation

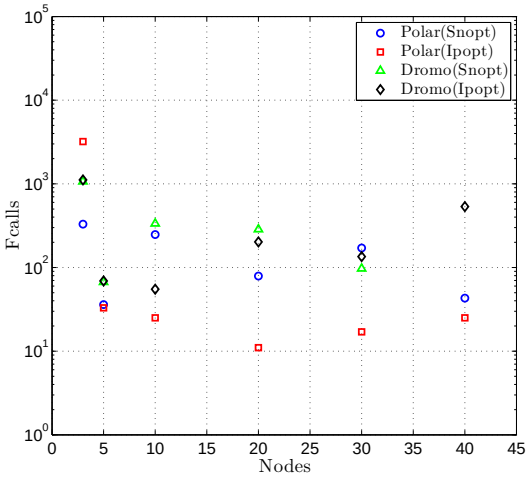
For the indirect Collocation case, only the Automatic Differentiation will be considered, as it has been proven to provide the more accurate and effective results. We obtain for this case a better precision using DROMO variables, whereas the computational cost is higher than when using Polar formulation. Solving the indirect problem with an NLP solver, shows to be not as accurate as the Direct approach, with the same order of iterations, runtime and Fcalls. As it has been seen, solving the Indirect problem do not worth while compared to the Direct method. The corresponding Data can be checked in Table (5.6).



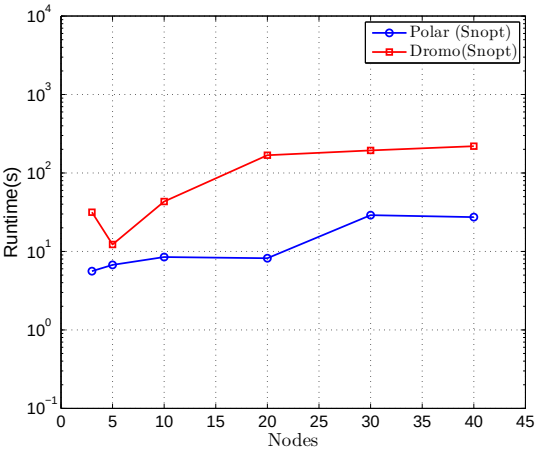
(A) Error vs N.nodes



(B) Iterations vs N.nodes



(C) F.calls vs N.nodes



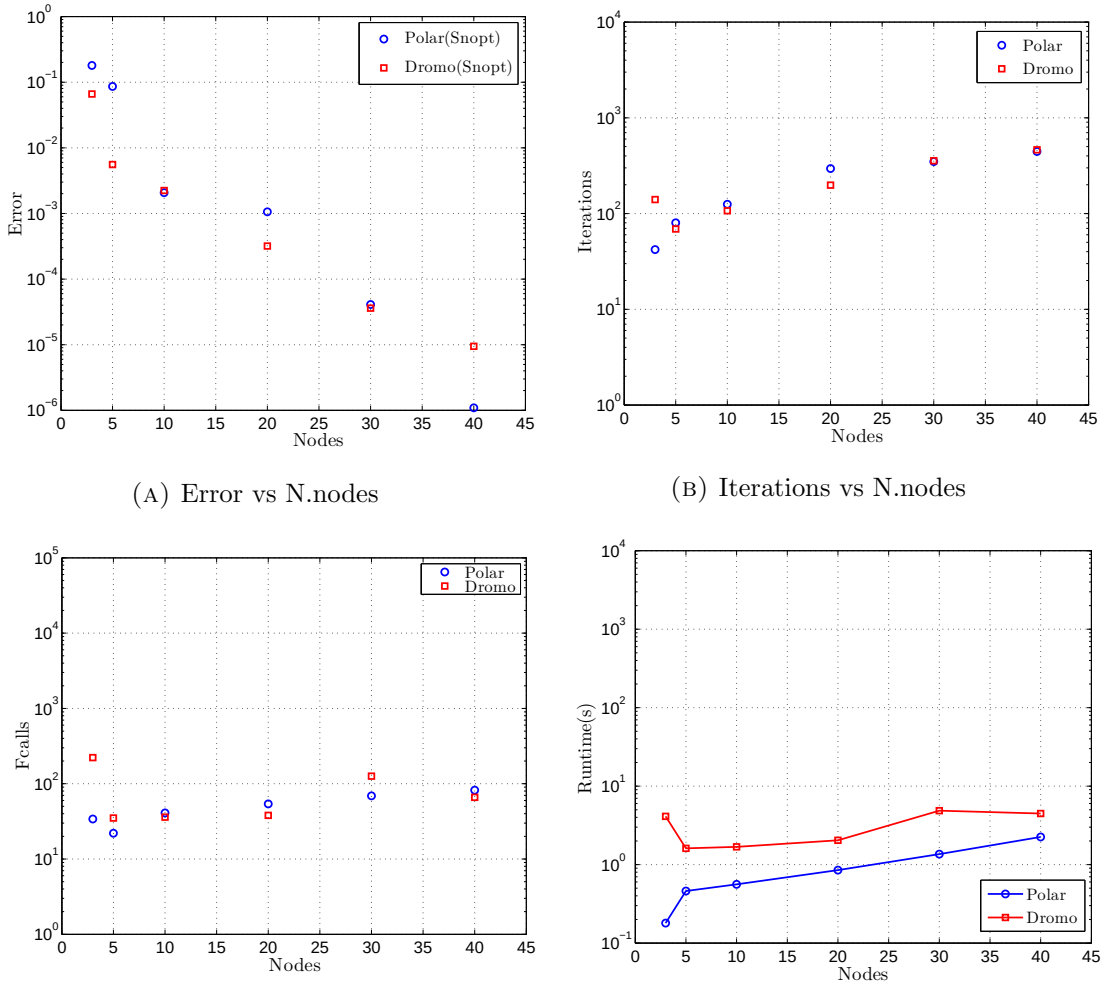
(D) Runtime vs N.nodes

FIGURE 5.6: SNOPT and IPOPT performance for Indirect Collocation method using Polar and Dromo variables

5.5 Numerical Solution by Pseudospectral Method

The results reported in this section have been obtained by using the implementation software $\mathcal{GPOPS}$, that uses the Gauss Pseudospectral Method and the Nonlinear solver SNOPT. In this test, Finite Differentiation, Automatic Differentiation ("object oriented") and Complex Differentiation will be compared for SNOPT. As it can be seen in Tab.(5.7) FD can provide accurate results when the number of variables is reduced. When more accurate results are demanded and more variables are used, they become inefficient, consuming a lot of time to compute gradients and not reaching the required precision. Complex Step differentiation can provide gradients with the precision required, almost up to machine precision produces it do not produce round-off errors.

Although it is more accurate than FD, it needs more time. Automatic Differentiation is proven to be the best option when a high number of variables is used consuming the lowest time. For the sake of clarity, only AD for DROMO and Polar performance are shown in Fig.(5.7)



(A) Error vs N.nodes

(B) Iterations vs N.nodes

(C) F.calls vs N.nodes

(D) Runtime vs N.nodes

FIGURE 5.7: SNOPT performance for Gauss Pseudospectral method using Polar and Dromo variables

5.6 Numerical Results: Tables

TABLE 5.4: SNOPT performance for linear approximation of the control in Polar and Dromo variables as a function of the number of nodes

Nodes	Variables	Solver	Infeas.	Iter.	F.calls	Error	Runtime(s)
3	Polar	SNOPT(FD)	2,50E-10	60	297	4,21E-03	3,80
3	Dromo	SNOPT(FD)	8,30E-04	117	927	1,82E-02	24,45
5	Polar	SNOPT(FD)	1,70E-09	124	1095	3,26E-04	9,28
5	Dromo	SNOPT(FD)	9,20E-03	439	1518	1,67E-02	37,23
10	Polar	SNOPT(FD)	1,30E-08	290	3465	6,38E-05	32,82
10	Dromo	SNOPT(FD)	5,50E-05	309	1408	1,43E-04	25,36
20	Polar	SNOPT(FD)	6,80E-03	940	2969	6,928E-03	28,50
20	Dromo	SNOPT(FD)	1,70E-02	2000	5665	1,941E-03	272,5
30	Polar	SNOPT(FD)	2,70E-02	1880	7647	1,476E-02	107,97
30	Dromo	SNOPT(FD)	9,00E-01	629	4147	4,027E-01	402,50

TABLE 5.5: SNOPT and IPOPT performance for Direct Collocation method using Polar and Dromo variables

Nodes	*Formulation*	*Solver*	*Infeas.*	*Iter.*	*F.calls*	*Error*	*Runtime(s)*
3	Polar	SNOPT(FD)	2,90E-08	75	790	1,73E-02	2,30
3	Polar	SNOPT(AD)	1,10E-07	95	163	1,73E-02	5,60
3	Polar	IPOPT(AD)	4,26E-07	38	44	1,73E-02	1,14
3	Polar	IPOPT(AD*)	4,26E-07	38	44	1,73E-02	11,52
3	Dromo	SNOPT(FD)	1,30E-04	339	4705	2,93E-02	2,30
3	Dromo	SNOPT(AD)	1,60E-06	485	1091	2,93E-02	31,58
3	Dromo	IPOPT(AD)	3,59E-00	576	7157	4,83E-01	19,12
5	Polar	SNOPT(FD)	1,20E-09	184	3617	3,93E-03	6,32
5	Polar	SNOPT(AD)	6,40E-08	200	339	3,93E-03	6,73
5	Polar	IPOPT(AD)	1,34E-07	63	79	3,93E-03	1,61
5	Polar	IPOPT(AD*)	1,34E-07	63	79	3,93E-03	6,81
5	Dromo	SNOPT(FD)	1,40E-05	2069	20922	3,06E-02	79,46
5	Dromo	SNOPT(AD)	1,20E-04	276	297	3,27E-03	12,26
5	Dromo	IPOPT(AD)	4,43E-07	313	768	1,02E-01	9,31
10	Polar	SNOPT(FD)	4,50E-06	203	7554	7,72E-04	29,34
10	Polar	SNOPT(AD)	3,80E-06	210	323	7,70E-04	8,49
10	Polar	IPOPT(AD)	5,00E-07	154	208	7,78E-04	4,56
10	Polar	IPOPT(AD*)	5,00E-07	154	208	7,78E-04	13,30
10	Dromo	SNOPT(FD)	1,70E-03	1753	41053	3,67E-04	570,83
10	Dromo	SNOPT(AD)	6,30E-12	1499	1202	8,19E-04	43,27
10	Dromo	IPOPT(AD)	1,46E-07	88	106	6,30E-03	8,31
20	Polar	SNOPT(FD)	1,10E-01	1100	10203	8,87E-03	114,62
20	Polar	SNOPT(AD)	2,50E-07	176	124	3,07E-05	8,19
20	Polar	IPOPT(AD)	3,08E-07	72	78	2,95E-05	2,36
20	Polar	IPOPT(AD*)	3,08E-07	72	78	2,95E-05	10
20	Dromo	SNOPT(AD)	2,00E-08	1001	2095	9,11E-06	168,67
20	Dromo	IPOPT(AD)	5,26E-07	213	297	1,78E-05	11,76
30	Polar	SNOPT(FD)	3,00E-13	475	23976	9,27E-03	667,48
30	Polar	SNOPT(AD)	2,30E-06	1007	275	2,28E-06	28,99
30	Polar	IPOPT(AD)	2,95E-07	29	30	1,30E-06	1,38
30	Polar	IPOPT(AD*)	2,95E-07	29	30	1,30E-06	7,39
30	Dromo	SNOPT(AD)	9,98E-07	1133	1200	9,72E-06	193,99
30	Dromo	IPOPT(AD)	2,32E-09	210	221	1,08E-05	14,77
40	Polar	SNOPT(AD)	9,86E-07	1018	589	9,86E-07	27,40
40	Polar	IPOPT(AD)	3,89E-07	24	26	2,34E-06	1,45
40	Polar	IPOPT(AD*)	3,89E-07	24	26	2,34E-06	10,82
40	Dromo	SNOPT(AD)	3,53E-07	1243	1345	4,10E-06	219,79
40	Dromo	IPOPT(AD)	2,09E-10	138	147	1,47E-06	13,93
100	Polar	IPOPT(AD)	2,48E-07	43	45	1,96E-06	7,63
100	Dromo	IPOPT(AD)	5,40E-09	34	35	2,21E-06	27,48

Table 5.6: SNOPT and IPOPT performance for Indirect Collocation method using Polar and Dromo variables

Nodes	*Formulation*	*Solver*	*Infeas.*	*Iter.*	*F.calls*	*Error*	*Runtime(s)*
3	Polar	SNOPT(AD)	2,80E-03	41	330	1,41E-02	5,4
3	Polar	IPOPT	1,25E-02	1000	3200	5,99E-02	34,99
3	Dromo	SNOPT(AD)	1,60E-06	485	1091	2,93E-02	31,58
3	Dromo	IPOPT	4,24E-00	500	1114	5,24E-01	49,25
5	Polar	SNOPT(AD)	1,50E-13	15	36	7,16E-04	5,60
5	Polar	IPOPT	3,25E-05	29	33	1,97E-03	1,40
5	Dromo	SNOPT(AD)	1,20E-04	276	297	3,27E-03	12,26
5	Dromo	IPOPT	7,49E-08	58	69	7,30E-02	7,84
10	Polar	SNOPT(AD)	6,40E-15	174	248	1,97E-03	13,11
10	Polar	IPOPT	3,01E-09	22	25	1,69E-03	0,93
10	Dromo	SNOPT(AD)	6,30E-12	1499	1202	8,19E-04	43,27
10	Dromo	IPOPT	5,95E-06	500	535	2,74E-02	399,35
20	Polar	SNOPT(AD)	5,40E-11	41	79	1,58E-03	18,65
20	Polar	IPOPT	8,13E-08	10	11	1,67E-03	0,65
20	Dromo	SNOPT(AD)	9,98E-07	1133	1200	9,72E-06	193,99
20	Dromo	IPOPT	4,09E-09	44	55	6,21E-04	7,36
30	Polar	SNOPT(AD)	1,80E-10	131	171	1,67E-03	36,48
30	Polar	IPOPT	3,18E-09	14	17	1,67E-03	1,06
30	Dromo	SNOPT(AD)	2,00E-08	1001	2095	9,11E-06	168,67
30	Dromo	IPOPT	3,38E-08	162	202	2,75E-05	28,98
40	Polar	SNOPT(AD)	1,90E-12	29	43	1,69E-03	66,45
40	Polar	IPOPT	3,27E-08	17	25	1,58E-03	1,67
40	Dromo	SNOPT(AD)	3,53E-07	1243	1345	4,10E-06	219,79
40	Dromo	IPOPT	4,53E-09	106	135	7,49E-06	31,05
100	Polar	SNOPT(AD)	X	X	X	X	X
100	Polar	IPOPT	3,65E-12	12	17	7,14E-04	8,79
100	Dromo	SNOPT(AD)	X	X	X	X	X
100	Dromo	IPOPT	5,52E-09	110	123	2,46E-06	829,75

TABLE 5.7: SNOPT performance for Gauss Pseudospectral method using Polar and Dromo variables

Nodes	*Variables*	*Solver*	*Infeas.*	*Iter.*	*F.calls*	*Error*	*Runtime(s)*
3	Polar	SNOPT(AD)	2,60E-10	42	34	1,796E-01	0,18
3	Polar	SNOPT(CD)	2,60E-10	42	34	1,796E-01	36,50
3	Polar	SNOPT(FD)	2,60E-10	42	456	1,796E-01	10,45
3	Dromo	SNOPT(AD)	2,50E-10	140	222	6,60E-02	4,12
5	Polar	SNOPT(AD)	2,00E-07	80	22	8,62E-02	0,46
5	Polar	SNOPT(CD)	2,00E-07	80	22	8,62E-02	31,52
5	Polar	SNOPT(FD)	2,00E-07	80	534	8,62E-02	9,19
5	Dromo	SNOPT(AD)	1,90E-08	69	35	5,56E-03	1,61
10	Polar	SNOPT(AD)	1,20E-09	125	41	2,07E-03	0,56
10	Polar	SNOPT(CD)	1,20E-09	125	41	2,07E-03	45,70
10	Polar	SNOPT(FD)	1,20E-09	125	3124	2,07E-03	12,44
10	Dromo	SNOPT(AD)	2,00E-08	107	36	2,23E-03	40,50
20	Polar	SNOPT(AD)	1,60E-07	295	54	1,06E-03	0,85
20	Polar	SNOPT(CD)	1,60E-07	295	54	1,06E-03	57,68
20	Polar	SNOPT(FD)	1,60E-07	284	4840	1,06E-03	24,14
20	Dromo	SNOPT(AD)	1,90E-12	198	38	3,18E-04	2,04
30	Polar	SNOPT(AD)	1,20E-07	349	69	4,10E-05	1,36
30	Polar	SNOPT(CD)	1,20E-07	349	69	4,10E-05	129,56
30	Polar	SNOPT(FD)	1,20E-07	394	11076	4,10E-05	65,78
30	Dromo	SNOPT(AD)	3,30E-07	356	126	3,60E-05	4,87
40	Polar	SNOPT(AD)	3,20E-07	445	82	1,09E-06	2,25
40	Polar	SNOPT(CD)	3,20E-07	445	82	1,09E-06	183,65
40	Polar	SNOPT(FD)	3,20E-07	485	18967	5,65E-05	121,43
40	Dromo	SNOPT(AD)	1,10E-07	463	66	9,45E-06	4,48

Chapter 6

Minimum Fuel-Consumption Orbit Transfer

6.1 Problem statement

The minimum fuel consumption orbit transfer problem implies the maximum final mass, which can be achieved by the most effective path between the initial and final orbit. This problem is significantly important given that, reducing the required mass propellant needed implies a reduction in the global cost of the mission. In this case, the problem can be formulated as follows:

"Given a constant-thrust rocket engine, T=thrust, operating for a given length of time, t_f, we wish to find the thrust-direction history, $\phi(t)$, to transfer a rocket vehicle from a given initial circular orbit to a given final circular orbit maximizing the final mass "

The nomenclature is defined in Sec.(5.1).

6.1.1 Polar formulation

This Optimal Control Problem can be formulated in terms of the set of Polar variables (r, u, v, t, m), where t represents the dimensional time, as

Find the control time-history $\phi(t)$ and the flight time t_f that maximizes $m_f = m(t_f)$.

The dynamics is governed by the first-order, two-dimensional equations of motion

$$\frac{dr}{dt} = u \tag{6.1}$$
$$\frac{du}{dt} = \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T\sin\phi}{m} \tag{6.2}$$
$$\frac{dv}{dt} = -\frac{uv}{r} + \frac{T\cos\phi}{m} \tag{6.3}$$
$$\frac{dm}{dt} = -|\dot{m}| \tag{6.4}$$

and must be solved given the *initial conditions*

$$m(t_0) = m_0, \qquad r(t_0) = r_0, \qquad u(t_0) = 0, \qquad v(t_0) = \sqrt{\mu/r(t_0)} \tag{6.5}$$

and subject to the *terminal constraints*

$$r(t_f) = r_f, \qquad u(t_f) = 0, \qquad v(t_f) = \sqrt{\mu/r_f} \tag{6.6}$$

Using the Calculus of Variations (Sec.2.2.2), the *Hamiltonian* has the following expression:

$$\mathcal{H} = \lambda_r u + \lambda_u(\frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T\sin\phi}{m}) + \lambda_v(-\frac{uv}{r} + \frac{T\cos\phi}{m}) - \lambda_m(|\dot{m}|) \tag{6.7}$$

$$\Phi = m(t_f) + \nu_1(r(t_f) - r_f) + \nu_2 u(t_f) + \nu_3[v(t_f) - \sqrt{\frac{\mu}{r_f}}] \tag{6.8}$$

Thus, the equations for the costates become

$$\frac{d\lambda_r}{dt} = -\lambda_u(-\frac{v^2}{r^2} + \frac{2\mu}{r^3}) - \lambda_v(\frac{uv}{r^2}) \tag{6.9}$$
$$\frac{d\lambda_u}{dt} = -\lambda_r + \lambda_v\frac{v}{r} \tag{6.10}$$
$$\frac{d\lambda_v}{dt} = -\lambda_u\frac{2v}{r} + \lambda_v\frac{u}{r} \tag{6.11}$$
$$\frac{d\lambda_m}{dt} = +\lambda_u(\frac{T\sin\phi}{m^2}) + \lambda_v(\frac{T\cos\phi}{m^2}) \tag{6.12}$$

and they are subject to the terminal conditions

$$\lambda_r(t_f) = \nu_1, \qquad \lambda_u(t_f) = \nu_2, \qquad \lambda_v(t_f) = \nu_3 \qquad \lambda_m(t_f) = 1 \tag{6.13}$$

Finally, the control can be obtained by using the Pontryagin's maximum (Ref.[8])principle as

$$0 = (\lambda_u\cos\phi - \lambda_v\sin\phi)\frac{T}{m_0 - |\dot{m}|t} \longrightarrow \tan\phi = \frac{\lambda_u}{\lambda_v} \tag{6.14}$$

Hence, the control can be determined in terms of λ_u and λ_v. Additionally, ν_1, ν_2, ν_3 should be chosen to satisfy the terminal conditions (Eq.(6.6)). Note that, maximizing the final mass consists on minimizing the final time, so that this case is a final time open problem.

6.1.2 DROMO formulation

The Trajectory Optimization Problem can be reformulated using DROMO variables in the particular planar case $[\zeta_1, \zeta_2, \zeta_3, \tau, m, \sigma]$, whose formulation is described in 3.5. Let us remark that the DROMO independent variable is the angular parameter σ instead of time t.

$$\tilde{a}_{px} = \frac{T\sin\phi}{m}/(\zeta_3^4 s^3), \qquad \tilde{a}_{py} = \frac{T\cos\phi}{m}/(\zeta_3^4 s^3) \tag{6.15}$$

The optimal control problem can be state as follows:

$$\textit{Find } \phi(t) \textit{ the flight time } \tau_f \textit{ that maximizes } m(\tau_f) = m_f$$

The system is governed by the following differential equations:

$$\frac{d\zeta_1}{d\sigma} = s\sin\sigma(\tilde{a}_{px}) + p(\tilde{a}_{py}) \tag{6.16}$$

$$\frac{d\zeta_2}{d\sigma} = -s\cos\sigma(\tilde{a}_{px}) + q(\tilde{a}_{py}) \tag{6.17}$$

$$\frac{d\zeta_3}{d\sigma} = -\zeta_3(\tilde{a}_{py}) \tag{6.18}$$

$$\frac{d\tau}{d\sigma} = \frac{1}{\zeta_3^3 s^2} \tag{6.19}$$

$$\frac{dm}{d\sigma} = -|\dot{m}|\frac{1}{\zeta_3^3 s^2} \tag{6.20}$$

and must be solved given the *initial conditions*

$$\zeta_1(\sigma_0) = 0, \qquad \zeta_2(\sigma_0) = 0, \qquad \zeta_3(\sigma_0) = 1/\sqrt{\mu r_0}, \qquad \tau(\sigma_0) = \tau_0 \tag{6.21}$$

and subject to the *terminal constraints*

$$\zeta_1(\sigma_f) = 0, \qquad \zeta_2(\sigma_f) = 0, \qquad \zeta_3(\sigma_f) = 1/\sqrt{r_f}, \tag{6.22}$$

The corresponding *Hamiltonian* can be formulated as

$$\mathcal{H} = \lambda_{\zeta_1}(s\sin\sigma(\tilde{a}_{px}) + p(\tilde{a}_{py})) + \lambda_{\zeta_2}(-s\cos\sigma(\tilde{a}_{px}) + q(\tilde{a}_{py})) \tag{6.23}$$

$$+\lambda_{\zeta_3}(-\zeta_3(\tilde{a}_{py})) + \lambda_\tau(1/(\zeta_3^3 s^2)) - \lambda_m(|\dot{m}|) \tag{6.24}$$

an

$$\Phi = m(\sigma_f) + \nu_1(\zeta_1(\sigma_f)) + \nu_2(\zeta_2(\sigma_f)) + \nu_3(\zeta_3(\sigma_f) - 1/\sqrt{r_f}) \tag{6.25}$$

Thus, the equation of the costates become

$$\frac{d\lambda_{\zeta_1}}{d\sigma} = -\lambda_{\zeta_1}[-\sin(2\sigma)\tilde{a}_{px} + (1+\cos^2(\sigma) - \frac{3}{s}p\cos\sigma)\tilde{a}_{py}] - \lambda_{\zeta_2}[(2\cos^2\sigma)\tilde{a}_{px}$$
$$+ (\sin\sigma\cos\sigma - \frac{3}{s}q\cos\sigma)\tilde{a}_{py}] - \lambda_{\zeta_3}[(\frac{3}{s}\zeta_3\cos\sigma)\tilde{a}_{py}] + (\lambda_\tau - |\dot{m}|\lambda_m)[\frac{2}{\zeta_3^3 s^3}\cos\sigma] \tag{6.26}$$

$$\frac{d\lambda_{\zeta_2}}{d\sigma} = -\lambda_{\zeta_1}[-2\sin^2(\sigma)\tilde{a}_{px} + (\sin\sigma\cos\sigma - \frac{3}{s}p\sin\sigma)\tilde{a}_{py}] - \lambda_{\zeta_2}[(\sin(2\sigma))\tilde{a}_{px}$$
$$+ (1+\sin^2\sigma - \frac{3}{s}q\sin\sigma)\tilde{a}_{py}] - \lambda_{\zeta_3}[(\frac{3}{s}\zeta_3\sin\sigma)\tilde{a}_{py}] + (\lambda_\tau - |\dot{m}|\lambda_m)[\frac{2}{\zeta_3^3 s^3}\sin\sigma] \tag{6.27}$$

$$\frac{d\lambda_{\zeta_3}}{d\sigma} = +\lambda_{\zeta_1}[s\sin\sigma(\tilde{a}_{px}) + p(\tilde{a}_{py})](\frac{4}{\zeta_3}) + \lambda_{\zeta_2}[-s\cos\sigma(\tilde{a}_{px}) + q(\tilde{a}_{py})](\frac{4}{\zeta_3})$$
$$- \lambda_{\zeta_3}[3\tilde{a}_{py}] + (\lambda_\tau - |\dot{m}|\lambda_m)[\frac{3}{\zeta_3^4 s^2}] \tag{6.28}$$

$$\frac{d\lambda_\tau}{d\sigma} = 0 \tag{6.29}$$

$$\frac{d\lambda_m}{d\sigma} = -(\lambda_{\zeta_1}(s\sin\sigma(\tilde{a}_{px}) + p(\tilde{a}_{py})) + \lambda_{\zeta_2}(-s\cos\sigma(\tilde{a}_{px}) + q(\tilde{a}_{py}))$$
$$+ \lambda_{\zeta_3}(-\zeta_3(\tilde{a}_{py})))(\frac{1}{m}) \tag{6.30}$$

The system is completed with the following *terminal adjoint conditions*:

$$\lambda_{\zeta_1}(\sigma_f) = \nu_1, \quad \lambda_{\zeta_2}(\sigma_f) = \nu_2, \quad \lambda_{\zeta_3}(\sigma_f) = -\nu_3, \quad \lambda_\tau(\sigma_f) = 0 \quad \lambda_m(\sigma_f) = 1 \tag{6.31}$$

Finally, the control can be determined as a function of the costates using the Pontryagin's maximum principle(Ref.[8]) as:

$$\lambda_{\zeta_1}(s\sin\sigma\cos\phi - p\sin\phi) + \lambda_{\zeta_2}(-s\cos\sigma\cos\phi - q\sin\phi) + \lambda_{\zeta_3}(\zeta_3\sin\phi) = 0 \tag{6.32}$$

$$\tan\phi = \frac{\lambda_{\zeta_1} s\sin\sigma - \lambda_{\zeta_2} s\cos\sigma}{\lambda_{\zeta_1} p + \lambda_{\zeta_2} q - \lambda_{\zeta_3}\zeta_3} \tag{6.33}$$

Note that, for DROMO formulation, the final state for the independent variable is unknown. Hence, it needs to be considered as an additional parameter which should be calculated during the optimization process.

6.2 Numerical Solution

The numerical results are intended to compare the performances between two different NLP solvers (SNOPT and IPOPT), as well as their comparative behaviour with the previous problem Maximum Radius (Chap.5). For this case, only the collocation and pseudospectral methods using Automatic Differentiation will be used, because they have proved effective for solving OCPs. The performances indexes were defined in Sec.(5.2) and we only need to redefine the benchmark solution. Given the relation between the maximum radius and maximum final mass, the maximum radius from the former r_f is chosen as the final radius orbit for the latter and the control solution must be the same for the continuous thrust problem. Hence, the difference between the established fuel consumption in Chap.(5) and the one obtained in the optimization process will state for the *Error* index. The remaining indexes are the same. For the sake of clarity in Tab.(6.1) all the numerical tests carried out are summarized.

TABLE 6.1: Numerical tests summary

Method	Problem	Variables	SNOPT			IPOPT	
			FD	AD	CD	AD	AD*
COLLOCATION	Direct	Polar		x		x	
		Dromo		x		x	
	Indirect	Polar		x		x	
		Dromo		x		x	
PSEUDOSPECTRAL	Direct	Polar		x			
		Dromo		x			

The initial guess for this problem consists on a linear interpolation between the initial and terminal state. The initial guess for the costates are set as equal to those obtained for the Maximum Radius Problem. The control is considered continuously tangent to the orbit, directed towards the velocity vector. The initial value for the unknown fuel consumption is set to zero.

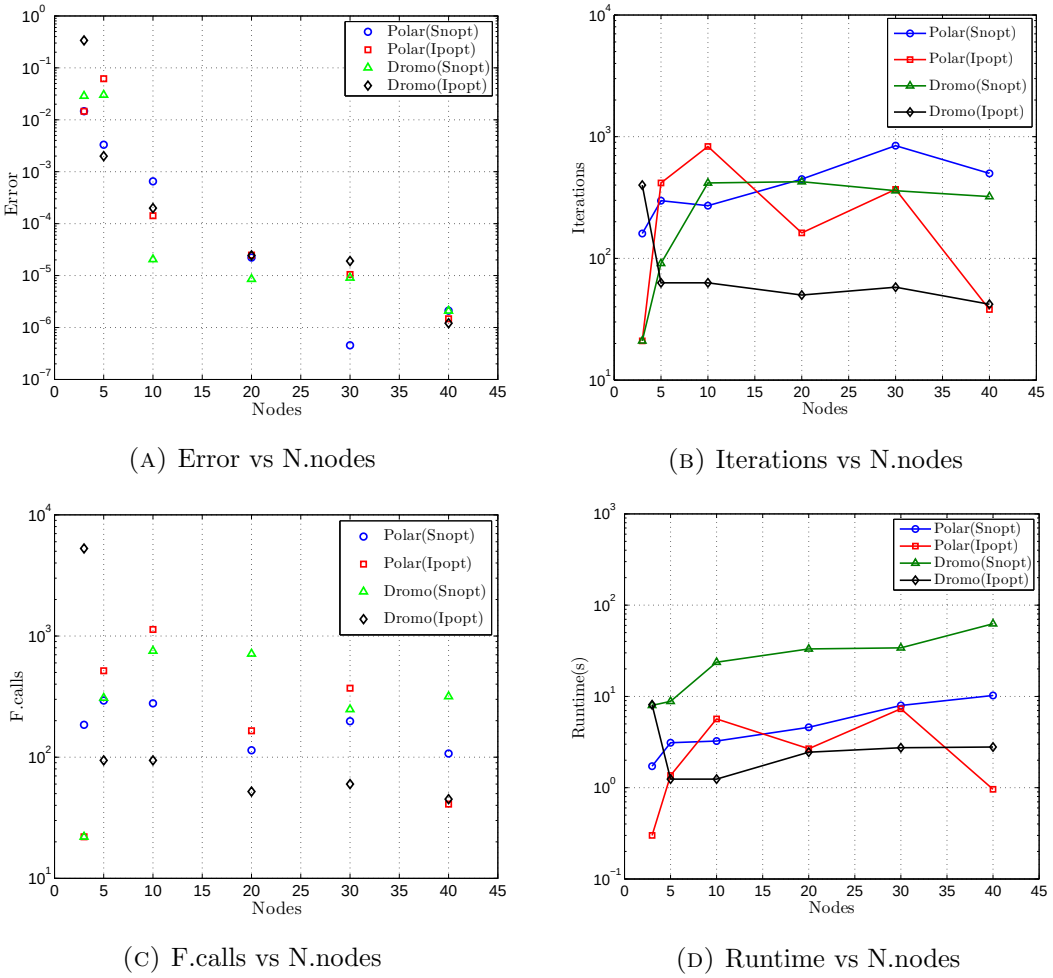
6.3 Numerical Solution by Collocation Method

In this section, the state/costate time history by a cubic Hermite interpolant as has been explained in (Sec.2.3.2.2), and the control will be piecewise-linear interpolated.

6.3.1 Direct Collocation

The performances can graphically be seen in Fig.(6.1) and the exact results are summarized in Tab.(6.2). The most accurate solution is obtained using SNOPT and Polar variables for $N = 30$. In general, IPOPT performs better than SNOPT requiring lower iterations and runtime than SNOPT. For this case, depending on the number of nodes chosen, DROMO can performed better than Polar.

Note that, using IPOPT with Polar variables has maximum Iterations, F.calls, and Runtime for $N = 10$. This is because the problem can reproduce better the optimal solution as the number of nodes increases, so that is less computational-costly for the solver to find the optimum.



(A) Error vs N.nodes

(B) Iterations vs N.nodes

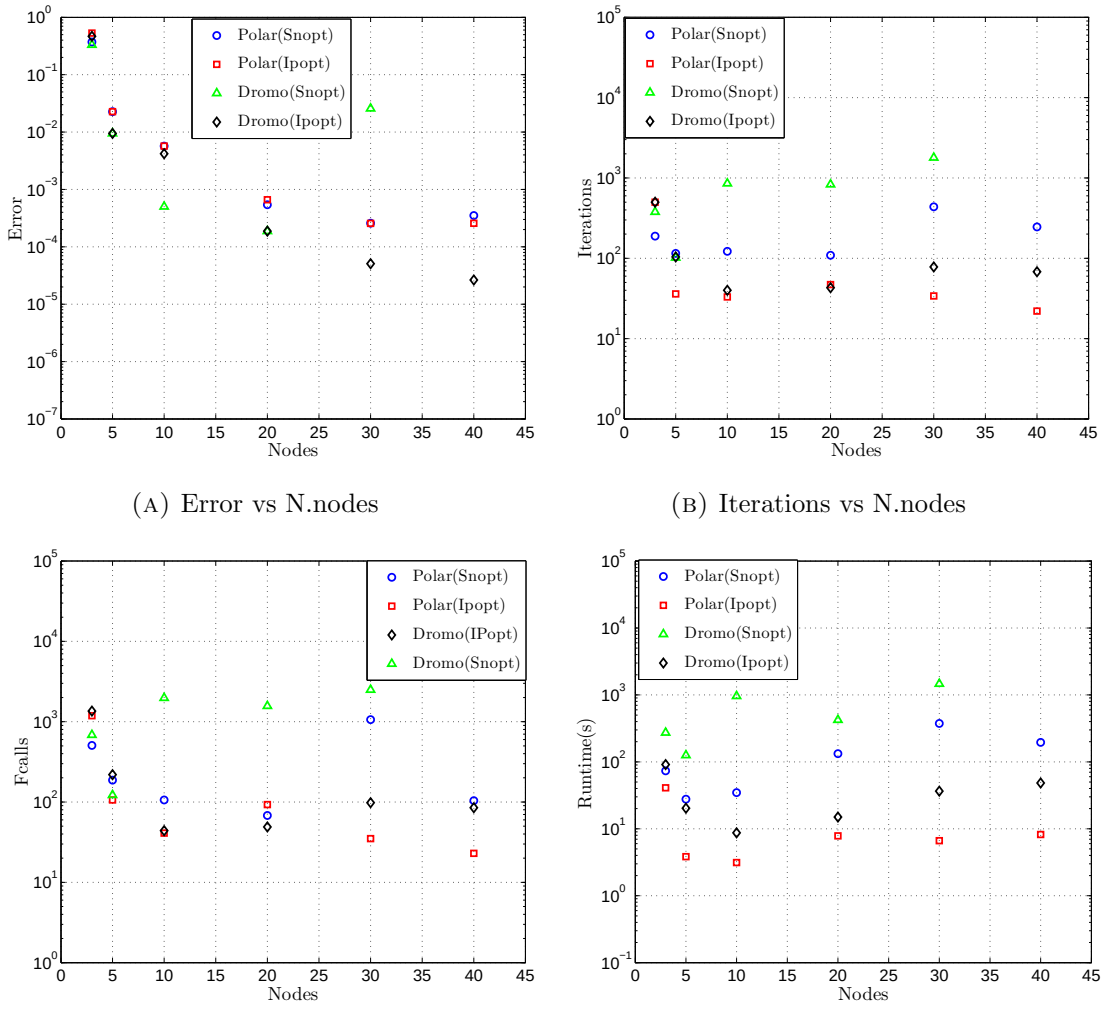
(C) F.calls vs N.nodes

(D) Runtime vs N.nodes

FIGURE 6.1: SNOPT and IPOPT performance for Direct Collocation method using Polar and Dromo variables

6.3.2 Indirect Collocation

Considering the indirect approach, DROMO formulation using IPOPT is seen to perform better than the others, obtaining the same precision than the direct method, with less computational-cost because the problem converges better. The problem using Polar variables, has an asymptotic solution, because it is not able to get an accuracy lower than 10^{-4}.



(A) Error vs N.nodes

(B) Iterations vs N.nodes

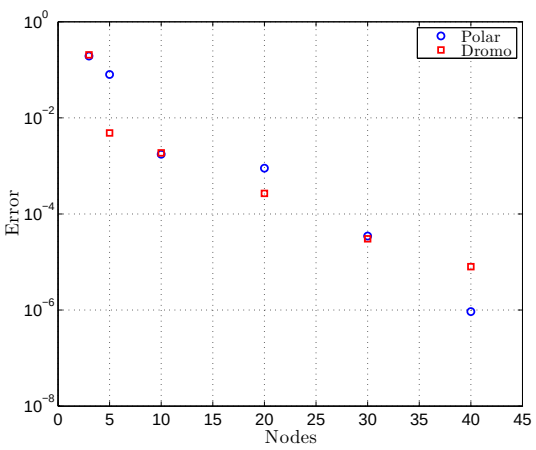
(C) F.calls vs N.nodes

(D) Runtime vs N.nodes

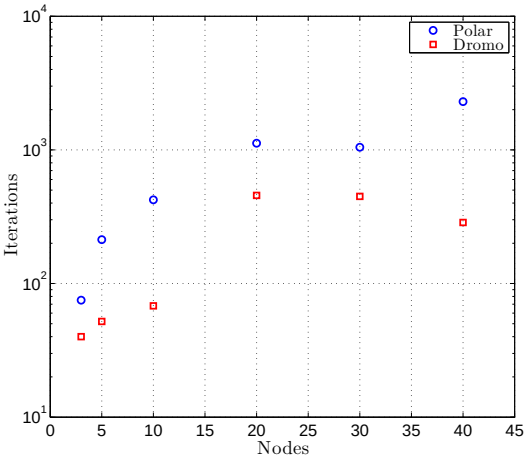
FIGURE 6.2: SNOPT and IPOPT performance for Indirect Collocation method using Polar and Dromo variables

6.4 Numerical Solution by Gauss Pseudospectral Method

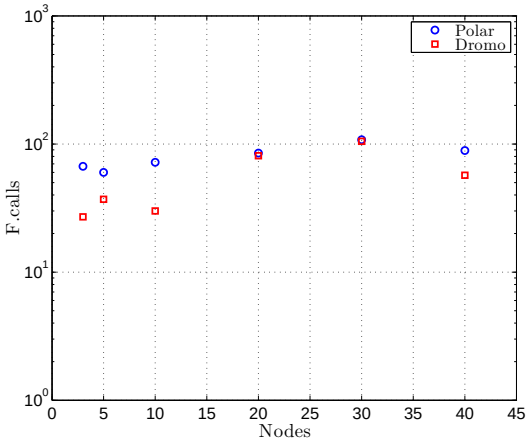
Using the $\mathcal{GPOPS}$ software implementation for SNOPT, the exact results are summarize in Tab.(6.4). As we can infer from Fig.(6.3) DROMO generally performs better than Polar variables. In general this method is slightly more straight-forward and robust than Direct Collocation.



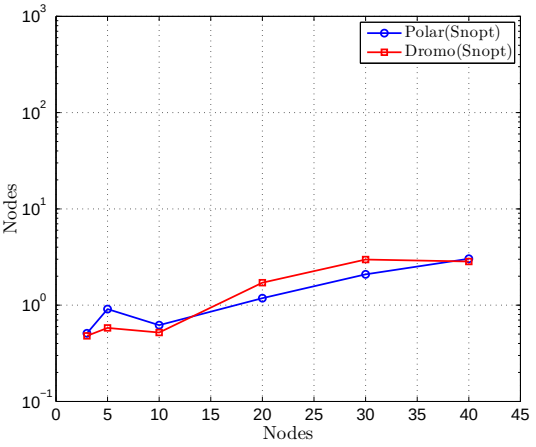
(A) Error vs N.nodes



(B) Iterations vs N.nodes



(C) F.calls vs N.nodes



(D) Runtime vs N.nodes

FIGURE 6.3: SNOPT performance for Gauss Pseudospectral method using Polar and Dromo variables

6.5 Numerical Results: Tables

TABLE 6.2: SNOPT and IPOPT performance for Direct Collocation method using Polar and Dromo variables

Nodes	*Formulation*	*Solver*	*Infeas.*	*Iter.*	*F.calls*	*Error*	*Runtime(s)*
3	Polar	SNOPT(AD)	5,10E-01	160	185	1,46E-02	1,72
3	Polar	IPOPT(AD)	3,50E-05	21	22	1,45E-02	0,30
3	Dromo	SNOPT(AD)	3,50E-05	21	22	2,87E-02	7,97
3	Dromo	IPOPT(AD)	5,60E-03	400	5272	3,38E-01	8,11
5	Polar	SNOPT(AD)	1,80E-09	298	293	3,30E-03	3,11
5	Polar	IPOPT(AD)	1,50E-04	417	516	6,18E-02	1,36
5	Dromo	SNOPT(AD)	1,50E-04	91	308	3,01E-02	8,82
5	Dromo	IPOPT(AD)	4,80E-10	63	94	1,99E-03	1,24
10	Polar	SNOPT(AD)	2,30E-08	271	278	6,52E-04	3,24
10	Polar	IPOPT(AD)	2,30E-07	831	1131	1,42E-04	5,67
10	Dromo	SNOPT(AD)	2,30E-07	417	753	2,05E-05	23,72
10	Dromo	IPOPT(AD)	1,80E-08	63	94	1,99E-04	1,24
20	Polar	SNOPT(AD)	3,60E-08	448	114	2,22E-05	4,59
20	Polar	IPOPT(AD)	2,40E-07	162	165	2,48E-05	2,67
20	Dromo	SNOPT(AD)	2,40E-07	427	713	8,57E-06	33,12
20	Dromo	IPOPT(AD)	1,30E-09	50	52	2,45E-05	2,45
30	Polar	SNOPT(AD)	4,40E-08	843	198	4,55E-07	7,95
30	Polar	IPOPT(AD)	6,60E-07	369	371	1,05E-05	7,33
30	Dromo	SNOPT(AD)	6,60E-07	360	248	9,07E-06	34,01
30	Dromo	IPOPT(AD)	3,30E-07	58	60	1,91E-05	2,74
40	Polar	SNOPT(AD)	8,20E-07	499	107	2,11E-06	10,24
40	Polar	IPOPT(AD)	4,50E-07	38	41	1,47E-06	0,96
40	Dromo	SNOPT(AD)	4,50E-07	322	317	2,09E-06	62,45
40	Dromo	IPOPT(AD)	2,90E-13	42	45	1,21E-06	2,79

Table 6.3: SNOPT and IPOPT Performance for Indirect Collocation method using Polar and Dromo variables

Nodes	*Formulation*	*Solver*	*Infeas.*	*Iter.*	*F.calls*	*Error*	*Runtime(s)*
3	Polar	SNOPT(AD)	5,60E-03	189	506	3,71E-01	73,43
3	Polar	IPOPT	5,35E-08	500	1188	5,31E-01	40,94
3	Dromo	SNOPT(AD)	2,20E+03	378	685	3,31E-01	273,30
3	Dromo	IPOPT	1,84E-01	500	1354	4,67E-01	91,09
5	Polar	SNOPT(AD)	4,80E-10	115	187	2,27E-02	27,73
5	Polar	IPOPT	3,72E-14	36	106	2,27E-02	3,83
5	Dromo	SNOPT(AD)	7,30E-07	102	122	9,47E-03	125,39
5	Dromo	IPOPT	1,73E-13	104	219	9,47E-03	20,30
10	Polar	SNOPT(AD)	1,80E-08	122	106	5,71E-03	34,76
10	Polar	IPOPT	2,43E-07	33	41	5,71E-03	3,12
10	Dromo	SNOPT(AD)	4,60E-09	855	1973	5,03E-04	959,85
10	Dromo	IPOPT	3,11E-12	40	44	4,19E-03	8,71
20	Polar	SNOPT(AD)	1,30E-09	109	68	5,42E-04	132,45
20	Polar	IPOPT	2,05E-13	47	93	6,62E-04	7,84
20	Dromo	SNOPT(AD)	7,60E-08	832	1567	1,88E-04	422,54
20	Dromo	IPOPT	1,59E-14	43	49	1,88E-04	14,99
30	Polar	SNOPT(AD)	3,30E-07	438	1060	2,58E-04	375,05
30	Polar	IPOPT	1,07E-08	34	35	2,56E-04	6,65
30	Dromo	SNOPT(AD)	2,30E-04	1792	2492	2,57E-02	1466,55
30	Dromo	IPOPT	1,65E-13	78	98	5,08E-05	36,57
40	Polar	SNOPT(AD)	2,90E-13	246	104	3,51E-04	195,68
40	Polar	IPOPT	1,24E-07	22	23	2,57E-04	8,21
40	Dromo	SNOPT(AD)	X	X	X	X	X
40	Dromo	IPOPT	6,49E-14	68	85	2,64E-05	48,26
100	Polar	SNOPT(AD)	X	X	X	X	X
100	Polar	IPOPT	5,35E-08	29	30	2,68E-04	57,91
100	Dromo	SNOPT(AD)	X	X	X	X	X
100	Dromo	IPOPT	9,08E-12	133	270	1,67E-05	730,53

TABLE 6.4: SNOPT performance for Gauss Pseudospectral method using Polar and Dromo variables

Nodes	*Formulation*	*Solver*	*Infeas.*	*Iter.*	*F.calls*	*Error*	*Runtime(s)*
3	Polar	SNOPT(AD)	1,80E-08	75	67	1,93E-01	0,51
3	Dromo	SNOPT(AD)	9,70E-08	40	27	2,04E-01	0,48
5	Polar	SNOPT(AD)	8,80E-10	213	60	8,00E-02	0,91
5	Dromo	SNOPT(AD)	1,00E-09	52	37	4,84E-03	0,58
10	Polar	SNOPT(AD)	7,60E-09	423	72	1,75E-03	0,62
10	Dromo	SNOPT(AD)	1,20E-10	68	30	1,87E-03	0,52
20	Polar	SNOPT(AD)	7,50E-08	1122	85	8,97E-04	1,18
20	Dromo	SNOPT(AD)	2,10E-09	456	81	2,68E-04	1,71
30	Polar	SNOPT(AD)	2,20E-08	1046	108	3,46E-05	2,09
30	Dromo	SNOPT(AD)	1,20E-08	449	105	3,03E-05	2,98
40	Polar	SNOPT(AD)	2,70E-07	2296	89	9,28E-07	3,03
40	Dromo	SNOPT(AD)	1,80E-08	286	57	7,98E-06	2,84

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this Project, we have summarized and understood the basic concepts underlying Optimal Control Problem solving. Capabilities to efficient solving more complicated problems have been developed given that Single Shooting, Collocation and Pseudospectral methods have been successfully implemented using Matlab. Moreover, advanced numerical tools and solvers used in this field have been correctly applied. The functionalities of the NLP solvers SNOPT and IPOPT have been exploited for both solving the direct and indirect approach. For most cases, the Interior Point solver IPOPT has performed better than the Sequential Quadratic Programming solver SNOPT in terms of of accuracy, convergence and computational cost.

The effectiveness of these techniques has been proven solving the classical brachistochrone problem, comparing the numerical solution with the existing analytical one. Then, the same software implementations have been applied to rightly solve a standard orbit transfer Optimization Problem, maximizing, in one case the final orbit radius, and, in the other case, maximizing the final mass.

In order to provide gradients to the NLP solvers different techniques have been considered:

- Finite Difference can provide accurate results when the number of variables is reduced. When more accurate results are demanded and more variables are used, they become inefficient, consuming a lot of time to compute gradients, or even leading to non-convergence.

- Complex Step differentiation can provide gradients with the precision required, almost up to machine precision produces it do not produce round-off errors. Although it is more accurate than FD, it needs more time.
- Automatic Differentiation is proven to be the best option when a high number of variables is used. They provide derivatives as accurate as analytical differentiation. The "object oriented" routines are the most efficient options, whereas "source code transformation" routines are more time consuming. However, the former is less flexible for coding than the latter.

From the tests performed, we can infer the following conclusions for the main Optimal Control methods:

- Single Shooting represents an easy implementable option to rapidly obtain moderate accurate solutions for both Direct and Indirect approaches. For this problem, the ODE solver should be carefully chosen to suit the specific problem.
- Collocation methods and Pseudospectral methods have been proven to offer high performances. Pseudospectral methods always need low computational time, but Direct Collocation methods can also obtain more accurate solutions in the same Runtime. One of this methods should be chosen depending on the specific problem. For this methods, the number of nodes should eb carefully chosen, since rising them generally yields the accuracy but also the computational cost.
- Solving the Indirect Method via NLP solvers does not generally improve the Direct approach, although sometimes it can converge faster because only the zero of a function must be found instead of solving an optimization problem. However, this does not overcome the need to analytically differentiate the equations for the costates and to raise the size of the problem.

In addition, the profit of using a different formulation of the dynamics for space trajectory optimization problems has been tested. Ideal frame-based high precision propagators have been explained comparing the performances between DROMO and DEPRIT, where DROMO has proven to perform better, and therefore has been chosen for the optimization problems. From the results obtained we can deduce that DROMO does not generally overtake the performance obtained using Polar variables. It can be explained by the following reasons:

- DROMO system involves more operations, which is a handicap for Automatic differentiation where the chain rule is applied and therefore more operations are performed.

- DROMO increases the order of the differential system, which generally suppose a burden for nonlinear solvers.
- DROMO is designed for low perturbed problems. The thrust-level considered in this project is not so small, as not even a half orbit is performed to transfer to the target orbit.

Overall, we can conclude that the optimal control techniques studied in this project have shown a strong potentiality in defining more efficient trajectories. Nevertheless, they need to be further exploited. Finally, choosing a method depends largely on the problem to be solved and the amount of time that can be invested in coding.

7.2 Future Work

Natural extensions of the applications solved in this project are threefold: First, to also consider more orbital perturbations such as the Earth Oblateness, the third body problem... Second, to considered "real" low-thrust problems where many revolutions around the Earth are needed to reach the target orbit. Third, improve ,optimize and innovate the existing techniques to effectively solve realistic problems.

Bibliography

[1] Peter Erickson. Spacecraft propulsion with impulsive performance analysis. 13, Stockholm 2011.

[2] John E. Prussing. Optimal impulsive linear systems: Sufficient conditions and maximum number of impulses. *Spaceflights Mechanics*, 1994. AAS 94.174.

[3] Claudio Bruno Paul A. Czysz. Enabling technologies for space exploration.

[4] Bruce A. Corway. Spacecraft trajectory optimization.

[5] Anil V Rao. *A survey of numerical methods for optimal control. Advances in the Astronautical Sciences*, 135(1):497?528, 2009.

[6] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193?207, 1998.

[7] William W.Hager Anil V Rao DAvid A. Benson Divya Garg, Michael Patterson and GEoffrey T. HUntington. A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica*, 46(11):1843–1851, 2010.

[8] L.S. Pontryagin. Mathematical theory of optimization processes. 1962.

[9] R.Weinstock. Calculus of variations. 1974.

[10] S.Wright J.Nocedal. Numerical optimization. 2006.

[11] Jesús Peláez, José Manuel Hedo, and Pedro Rodriguez de Andrés. A special perturbation method in orbital dynamics. *Celestial Mechanics and Dynamical Astronomy*, 97:131–150, 2007. ISSN 0923-2958. 10.1007/s10569-006-9056-3.

[12] J.Zabczyk. Mathematical control theory. 2008.

[13] *Applied Optimal Control: Optimization, Estimation, and Control.* Taylor and Francis, 1975.

[14] R.D. Russell U.M. Ascher, R.M. Mattheij. Numerical solution of boundary-value problems in ordinary differential equations.

[15] J.T. War Kman S.Lehnart. Optimal control applied to biologic models. 2007.

[16] H.B. Keller. Numerical solution of two point boundary value problems. 1976.

[17] W.C.Gear. Numerical initial-value problems in ordinary differential equations. 1971.

[18] Anil V. Rao. *User's Guide for GPOPS Version 2.3 : Software for solving Multiple-Phase Optimal Control Problem Using the Gauss Pseudospectral Methods*, institution = University of Florida, Gainesville, FL 32607, year = August,2009,. Technical report.

[19] Philip R. GILL. *User's Guide for SNOPT Version 7 : Software for Large-Scale Nonlinear Programming.* Technical report, Department of Mathematics, University of California, San Diedo, La Jolaa CA 92093-0112, June 16,2008.

[20] Andreas Wachter Stefan Vigerske. *Introduction to Ipopt: A tutorial for downloading, installing and using Ipopt.* Technical report, Carnegie Mellon University, April 8,2014.

[21] A.Deprit. Ideal elements for perturbed keplerian motions. *Journal of Research of the National Bureau of Standards.Sec.B: Math. Sci*, 79B(1-2):1–15, 1975. Paper 79B1&2-416.

[22] P.A.Hansen. Auseinandersetzung einer zweckmessigen methode zur berechnung der aboluten storungen der kleinen planeten. *ABh.der Math.-Phys. Cl.der Kon. Sachs. Ges. der Wissenschaft*, 5:42–218, 1857.