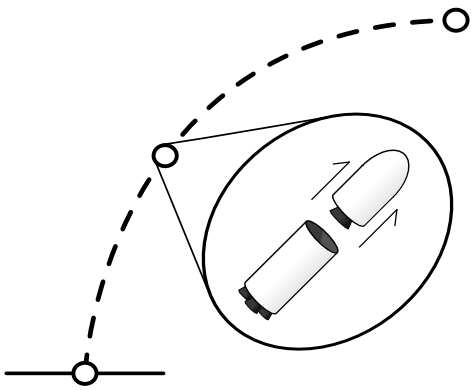
TÉCNICO LISBOA



Development of Multiphase Radau Pseudospectral Method for Optimal Control Problems

José Eduardo Valério Garrido

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. Paulo Jorge Soares Gil
Dr.-Ing. Marco Sagliano

Examination Committee

Chairperson: Prof. Filipe Szolnoky Ramos Pinto Cunha
Supervisor: Prof. Paulo Jorge Soares Gil
Member of the Committee: Prof. Bruno João Nogueira Guerreiro

January 2021

Aos meus irmãos

Alice e Luis

Authorship Declaration

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Agradecimentos

Quero agradecer à DLR, nomeadamente, ao departamento GNC do Instituto de Sistemas Espaciais em Bremen pela oportunidade fantástica que me foi concedida, sem a qual este trabalho não teria sido possível.

Um obrigado especial vai para o Dr. Marco Sagliano pela sua fé em mim e pela sua orientação, sem as quais eu encontrar-me ia perdido. O Dr. Sagliano foi a verdadeira mente por de trás deste trabalho, não consigo agradecer-lhe o suficiente.

Quero agradecer também ao Professor Paulo Gil, não só por me apoiar neste empreendimento, mas também pela sua orientação e pelos seus conselhos imprescindíveis.

Um grande obrigado à minha querida amada Carmen Machado pelo seu suporte, pela sua motivação e pelo seu humor, essenciais nos tempos mais difíceis.

Finalmente, eu quero agradecer profundamente à minha mãe, Deolinda Valério, ao meu pai, Eduardo Garrido, à minha tia, Elisa Valério, e aos meus avós Maria e Manuel Valério pelo seu suporte e atenção contínuos durante toda a minha vida académica.

Acknowledgements

I want to thank the DLR, namely, the GNC department at the Institute of Space Systems in Bremen, for providing me with this amazing opportunity without which this work would not have been possible.

A special thanks goes to Dr. Marco Sagliano for his faith in me and his ceaseless guidance, without which I would find myself lost. Dr. Sagliano was the true mastermind behind the writing of this thesis. I cannot thank him enough.

I also want to thank Professor Paulo Gil not only for endorsing me in this quest, but also for his priceless advice and persistent reality checks.

A big thank you to my beloved sweetheart Carmen Machado for her support, motivation and dry humour, getting me through the most difficult times.

Finally, I want to deeply thank my mother, Deolinda Valério, my father, Eduardo Garrido, my aunt, Elisa Valério, and my grandparents, Maria and Manuel Valério, for their continuous care and support throughout my academic life.

Resumo

Neste trabalho, o método pseudoespectral de Radau invertido é usado para resolver problemas de controlo óptimo compostos de múltiplas fases. A aplicação de interesse é a geração de trajectórias de foguetes compostos de múltiplos estágios em ambos os contextos de ascensão para órbita e de descida para aterragem vertical. O método é implementado no programa SPARTAN, desenvolvido em MATLAB pelo Centro Aeroespacial Alemão (DLR), que utiliza o *solver* de programação não linear IPOPT. São implementados dois problemas numéricos relevantes para validar o algoritmo. Os problemas são apropriados porque, de uma fase para a seguinte, os respectivos sistemas dinâmicos estão sujeitos ou a alterações nas equações do movimento ou a descontinuidades de salto nos seus estados (ou ambos). O primeiro problema é composto de três fases distintas e baseia-se na recuperação do estágio principal do lançador orbital *Falcon 9* através de uma manobra de impulso de retorno e de aterragem vertical. O segundo problema é composto de quatro fases e baseia-se no lançamento, ascensão e inserção em órbita do foguete orbital de múltiplos estágios *Delta III*. O algoritmo é validado através da comparação directa dos resultados com as respectivas fontes e também através da análise ao Hamiltoniano e às variáveis complementares associadas. Os resultados mostram que o método é capaz de gerar trajectórias óptimas com exactidão comparável a outros programas de última geração.

Palavras Chave

controlo óptimo, método pseudoespectral de Radau, trajectória de foguetes multiestágios, aterragem vertical, trajectória ascendente, inserção em órbita.

Abstract

In this work, the flipped Radau pseudospectral method is employed in order to solve multiphase optimal control problems. The application of interest is the generation of multi-stage rocket trajectories, both in ascent to orbit and descent to vertical landing. The method is implemented in the MATLAB tool SPARTAN, developed at DLR, with the use of the NLP solver IPOPT. Two relevant numerical examples are implemented for validation of the algorithm. The examples are appropriate because they contain either mutations in the equations of motion from one phase to the next or jump discontinuities in the states (or both). The first problem is composed of three distinct phases and is concerned with the recovery of the main booster of the orbital launcher Falcon 9 via a boost-back manoeuvre and a vertical landing near the launch site. The second problem is composed of four phases and is concerned with the launch, ascent and orbit insertion of the multiple-staged solid fuel orbital rocket Delta III. The algorithm is validated both, by the comparison of the results to the corresponding reference solutions, and by the analysis of the dual variables and of the Hamiltonian associated with the trajectories. The results show that the method is able to generate optimal trajectories with accuracy comparable to state of the art solvers.

Keywords

optimal control, Radau pseudospectral method, multi-stage rocket trajectory generation, boost-back and vertical landing, ascent trajectory, orbit insertion.

Contents

List of Tables

List of Figures

List of Symbols

Roman symbols

$\boldsymbol{C}(\boldsymbol{X}_{\text{NLP}})$ Algebraic constraints of the nonlinear programming problem

$\boldsymbol{D}$ Chapters 2, 3 and Appendix A: Differentiation matrix. Chapter 4: Drag force vector

$\boldsymbol{D}^*$ Dual differentiation matrix

$\boldsymbol{F}$ Array of concatenated dynamic defects

$\boldsymbol{f}(\cdot)$ Generic algebraic vector function representing the right-hand side of the equations of motion

$\boldsymbol{F}_{\text{ext}}$ External forces which do not change the mass of the system

$\boldsymbol{F}_{\text{net}}$ Total net force acting on the system

$\boldsymbol{g}$ Vector of gravitational acceleration of the Earth

$\boldsymbol{H}$ Array of concatenated path constraints

$\boldsymbol{h}(\cdot)$ Generic algebraic vector function representing path constraints

$\boldsymbol{I}_{n\times n}$ Identity matrix of dimension n by n

$\boldsymbol{r}$ Spacecraft position vector

$\boldsymbol{T}$, T Thrust vector, thrust vector norm

$\boldsymbol{u}(t)$ Generic control vector

$\boldsymbol{v_e}$, v_e Exahust velocity vector, exahust velocity vector norm

$\boldsymbol{v}$ Spacecraft velocity vector

$\boldsymbol{x}(t)$ Generic state vector

$\boldsymbol{X}_{\text{NLP}}$ Vector of decition variables of the nonlinear programming problem

$\boldsymbol{\ell}$ Algebraic vector function representing linkage conditions

$\mathcal{H}$ Variational Hamiltonian

$\mathcal{H}^\dagger$ Scalar constant representing the optimal Hamiltonian

$\mathcal{J}$ Performance measure expressed as a cost functional

$\mathcal{J}^{\lambda N}$ Augmented-discrete cost functional

$\mathcal{J}^{\lambda}$ Augmented cost functional

$\mathcal{J}^{N\lambda}$ Discrete-augmented cost functional

$\mathcal{J}^{N}$ Discrete cost functional

$\mathcal{L}$ Lagrangian of the Hamiltonian

$\|\boldsymbol{x}\|$	Vector Euclidean norm
MR	Vehicle Mass ratio
$\veebar$	Logic operator for exclusive disjunction
$\widetilde{\mathcal{H}}$	KKT Hamiltonian
$\widetilde{\mathcal{L}}$	KKT Lagrangian
C_D	Drag coefficient
$C_{\lambda,k}$	Scalar factor of the dynamic defects corresponding to the k^{th} sample point along the time domain
$C_{\mu,k}$	Scalar factor of the path constraints corresponding to the k^{th} sample point along the time domain
g_0	Gravitational acceleration of the Earth at the surface
h_0	Density scale height
I_{sp}	Engine specific impulse
$J(\boldsymbol{X}_{\text{NLP}})$	Cost function of the nonlinear programming problem
m	Vehicle mass
$n_{\boldsymbol{x}}$	Dimension of vector $\boldsymbol{x}$, such that $\boldsymbol{x} \in \mathbb{R}^{n_{\boldsymbol{x}}}$
$P_N(\tau)$	Legendre polynomial of order N
$R_N(\tau)$	Flipped Radau polynomial of order N
t	Independent variable for the time domain
t_0	Initial time domain point
t_f	Final time domain point
w_k	Gauss quadrature weight corresponding to the k^{th} sample point along the domain

Greek symbols

$\boldsymbol{\lambda}$	Covector associated with dynamic defects
$\boldsymbol{\mu}$	Covector associated with path constraints
$\boldsymbol{\nu}$	Covector associated with event constraints
$\boldsymbol{\phi}(\cdot)$	Generic algebraic vector function representing event constraints
$\boldsymbol{\xi}$	Dynamic defect constraints
$\Delta\boldsymbol{x}$	Instantaneous change in state vector for linkage conditions
Δt	Period of time
ΔV	Change in velocity
μ	Standard gravitational parameter of the Earth
Φ	Mayer term of the cost functional
Ψ	Lagrange term of the cost functional
ρ	Atmospheric air density
ρ_0	Standard atmospheric air density at sea level
τ	Normalized domain of the independent variable
τ^{DR}	Abscissas corresponding to the roots of the direct Radau polynomial

$\tilde{\boldsymbol{\lambda}}$ KKT multiplier associated with dynamic defects

$\tilde{\boldsymbol{\mu}}$ KKT multiplier associated with path constraints

$\tilde{\boldsymbol{\nu}}$ KKT multiplier associated with event constraints

δ Variation operator

ζ Propellant mass fraction

Subscripts

0 Value of vector at the initial point of the time domain

max Upper boundary vector

min Lower boundary vector

f Value of vector at the final point of the time domain

i Value of vector at the i^{th} sample point along the domain

k Value of vector at the k^{th} sample point along the domain

ki Matrix element in the k^{th} row and i^{th} column

Superscripts

(p) Vector in phase p

T Vector transpose

List of Abbreviations

CMT	Covector Mapping Theorem.
DLR	Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Centre).
ECI	Earth-Centered Inertial reference frame.
EoM	Equations of Motion.
HBVP	Hamilton Boundary-Value Problem.
ISS	International Space Station.
KKT	Karush–Kuhn–Tucker optimality conditions.
NLP	Non-linear Programming.
OCP	Optimal Control Problem.
ODE	Ordinary Differential Equation.
OLV	Orbital Launch Vehicle.
SPARTAN	Simple Pseudospectral Algorithm for Rapid Trajectory ANalysis.

Chapter 1

Introduction

1.1 Objective

In this thesis, the flipped Radau Pseudospectral method for optimal control is extended to solve multi-phase problems. This extension of the algorithm allows the study of problems that might contain state and/or control discontinuities or local mutations in the equations of motion, thus augmenting in complexity and in quantity the range of problems that are possible to solve. The method is applied to the case of multiple stage launch vehicles in both ascent from surface to orbit and descent from free-fall to vertical landing.

1.2 The Orbital Launch Vehicle

An orbital launch vehicle (OLV) is a machine designed to take payloads to a specified orbit in space. These machines, commonly known as *Rockets*, play a very important role in implementing (and sometimes maintaining) modern world utilities such as Global Positioning Systems, telecommunication services, space observatories, among others. Therefore, it is of ultimate interest for these machines to perform in a way that is fuel-effective and which allows a safe payload delivery.

This is not a trivial task, mainly because the propulsion system of a rocket requires high values of propellant mass fraction, which is to say that the propellant alone makes up a significant proportion of the total mass of the vehicle. Essentially, the rocket has to transport all of the fuel required to propel itself along the trajectory, but the more fuel it carries the more fuel is required due to the increased weight. The *Tsiolkovsky Rocket equation* can be used in order to calculate the propellant mass fraction of the vehicle given a required velocity budget.

1.2.1 Tsiolkovsky Rocket Equation and Staging

The *Tsiolkovsky rocket equation* provides a scalar evaluation of the performance of a given vehicle configuration. In the absence of external forces, such as gravity or aerodynamic drag, and assuming perfect fuel consumption, there is a limit to how much velocity a rocket can gain. This limit is related to the mass ratio of

the vehicle by [1]

$$\Delta V = v_e \ln(\mathrm{MR}) = -v_e \ln(1-\zeta)\,, \tag{1.1}$$

where ΔV is, for a given configuration, the available change in velocity the rocket acquires and v_e is the effective exhaust velocity. The mass ratio, MR, is the quotient between the initial and final mass of the vehicle, and ζ is the propellant mass fraction as follows [1]:

$$\mathrm{MR} = \frac{\text{Initial Mass}}{\text{Mass at Burnout}}\,, \tag{1.2}$$

$$\zeta = \frac{\text{Propellant Mass}}{\text{Initial Mass}} = 1 - \frac{1}{\mathrm{MR}}\,. \tag{1.3}$$

The higher ΔV a vehicle configuration has, the better it is expected to perform. Figure 1.1 illustrates the relationship between the propellant mass fraction, ζ, and the ratio of output ΔV to exhaust velocity, v_e, given by (1.1). The asymptotic gradient of the plot makes it clear that achieving high values of ΔV is not trivial, as this implies that the propellant mass fraction will tend to 1 — a vehicle with no room for payload or structural mass. In practice, the values of propellant mass fraction typically lie in between 0.75 to 0.95 [2].

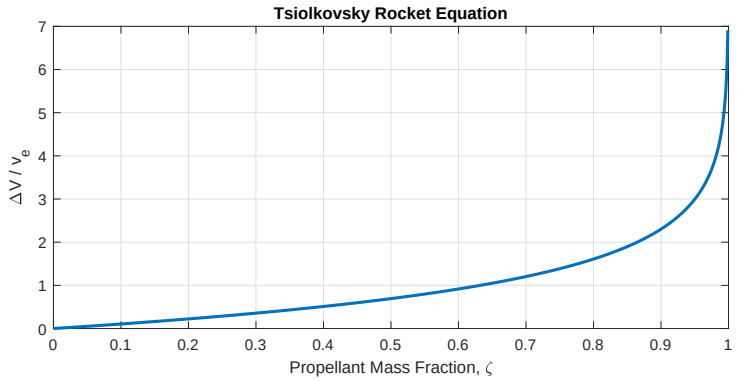


Figure 1.1: The Tsiolkovsky rocket equation relates the propellant mass fraction, ζ, to the ratio $\Delta V/v_e$.

One ingenious way to work around this is the concept of *multi-stage rocket*. As expressed in (1.4), a multi-stage rocket allows the partition of the total ΔV into multiple contributions. Letting P be the number of stages of the vehicle, the rocket equation becomes [3]

$$\Delta V = \sum_{p=1}^{P} v_e^{(p)} \ln\left(\mathrm{MR}^{(p)}\right) = -\sum_{p=1}^{P} v_e^{(p)} \ln\left(1-\zeta^{(p)}\right)\,. \tag{1.4}$$

A multi-stage rocket is advantageous because it makes it possible to dispose of inert mass along the trajectory as empty burnout stages are detached and jettisoned in stage separation events. At the cost of ejecting fully functioning propulsion systems along with empty tanks and structural mass at stage separation events, a multi-stage configuration also allows the modification of engine specification from stage to stage, therefore the propulsion system of each stage can be tailored to meet the requirements of the environment it will be working in. It is relevant to note that if not for the detachment of the empty stages, the multiphase configuration would be almost redundant due to the additive property of the logarithmic function.

The difficulty in achieving high values of ΔV due to the character of the rocket equation is only aggravated when the effects of gravity and atmospheric drag are taken into account. Both of these forces act in opposition

to the movement of the rocket during the launch sequence, which results in the suppression of the total ΔV budget of the vehicle.

In the case of gravity, for instance, assuming a vertical launch and a constant gravitational acceleration, a rough estimate of the velocity budget can be computed based on the duration of flight, Δt as follows [4]:

$$\Delta V_{\text{Actual}} = \Delta V_{\text{Total}} - \Delta V_{\text{Gravity}} \,, \tag{1.5}$$

$$\Delta V_{\text{Gravity}} = g_0 \Delta t \,, \tag{1.6}$$

where ΔV_{Actual} is the expected velocity budget of the vehicle when subject to the force of gravity, ΔV_{Total} is the velocity budget of the rocket calculated via (1.4) (in isolation of external forces), $\Delta V_{\text{Gravity}}$ is the equivalent velocity expense of gravity and g_0 is the standard gravity acceleration of the Earth.

Ultimately, a staged rocket makes it easier to achieve high values of ΔV_{Total}, necessary to overcome gravity and atmospheric drag expenses. The motivation for multi stage rockets is thus established.

1.2.2 Equations of Motion

Generic formulation

As a dynamic system, an OLV a can be represented by a state vector, $\boldsymbol{x}(t)$, descriptive of relevant properties associated with the system. In this case, the state vector will often consist on the group *position-velocity-mass* [5–7].

The behaviour of the vehicle is described by the equations of motion (EoM) which describe the rate of change of the state vector with time. A sequence of states associated with a time interval is called a *trajectory* and it is possible to influence the trajectory by introducing the control vector, $\boldsymbol{u}(t)$, into the equations of motion. In the case of an OLV, the control vector will often consist on the components of *thrust*.

Given the state and control vectors, respectively $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$, where n_x and n_u are the respective vector dimensions, it is said that the system is subject to equations of motion, generically represented as [8]:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \,, \tag{1.7}$$

where $\boldsymbol{f}(\cdot)$ is a vector function representative of the right-hand side of the differential equations. The vector function $\boldsymbol{f}(\cdot)$ can be explicitly dependent on the state of the system, $\boldsymbol{x}(t)$ and on the specified control vector, $\boldsymbol{u}(t)$. The generic formulation of (1.7) will be used throughout this work as a way to express any given dynamic system.

Equations of motion for rockets

The propulsive system of a rocket is based on the reactive force from mass exhaust. The equations of motion for a rocket modelled as a point mass can be expressed as follows [3]:

$$\boldsymbol{F}_{\text{net}} = \boldsymbol{F}_{\text{ext}} + \boldsymbol{T} = m\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} \,, \tag{1.8}$$

$$\boldsymbol{T} = \boldsymbol{v}_e \frac{\mathrm{d}m}{\mathrm{d}t} \,, \tag{1.9}$$

where the vector $\boldsymbol{F}_{\text{net}}$ is the resulting net force acting on the vehicle, the vector $\boldsymbol{F}_{\text{ext}}$ is the sum of common external forces that do not change the mass of the system, $\boldsymbol{T}$ is the thrust vector, m is the total mass of the vehicle, $\boldsymbol{v}$ is the velocity vector, and $\boldsymbol{v}_e$ is the effective mass exhaust velocity vector. Equations (1.8) and (1.9) account for propulsion based on both escaping mass, $\frac{\mathrm{d}m}{\mathrm{d}t} < 0$, and incident mass, $\frac{\mathrm{d}m}{\mathrm{d}t} > 0$, with respect to the center of mass of the vehicle. Because the mass flow rate, $\frac{\mathrm{d}m}{\mathrm{d}t}$, is a scalar quantity, it yields that the thrust vector, $\boldsymbol{T}$, is always collinear with the exhaust velocity vector, $\boldsymbol{v}_e$.

The equations of motion for rockets and other rocket-like propulsive systems can be written in state-space representation without loss of generality by solving (1.8) and (1.9) for the rates of change of velocity and mass, respectively, $\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}$ and $\frac{\mathrm{d}m}{\mathrm{d}t}$, and adjoining the kinematic equations $\left(\dot{\boldsymbol{r}} = \boldsymbol{v}\right)$, where $\boldsymbol{r}$ is the position vector:

$$\dot{\boldsymbol{r}} = \boldsymbol{v}\,, \tag{1.10}$$

$$\dot{\boldsymbol{v}} = \frac{\boldsymbol{F}_{ext}\left(\boldsymbol{r}, \boldsymbol{v}\right)}{m} + \frac{\boldsymbol{T}}{m}\,, \tag{1.11}$$

$$\dot{m} = -\frac{\left\|\boldsymbol{T}\right\|}{v_e}\,, \tag{1.12}$$

where the generic force $\boldsymbol{F}_{ext}(\cdot)$ is representative of external sources of acceleration such as gravitational forces, aerodynamic forces, or fictitious forces introduced by a non-inertial reference frame, and the norm of the effective exhaust velocity vector, $v_e = \|\boldsymbol{v}_e\|$, is a known constant for a given vehicle.

Equations (1.10) to (1.12) presume a point-mass approximation of the system, this is to say that these equations overlook some aspects of rocket behaviour. For instance, this model does not encompass the attitude control of the vehicle, and therefore the angle of attack is assumed to be zero at all times. More refined approximations imply the extension of the state vector and the consequent appending of additional equations of motion.

1.2.3 Trajectory Optimization

Before every launch, a trajectory should be generated in order to minimize the fuel consumption of the vehicle. This is where the discipline of optimal control comes in.

Optimizing multi-staged rocket trajectories is not an easy task. The difficulties include:

- The nonlinear nature of the equations of motion.
- The modelling of physical phenomena, such as drag which is a function of the relative air velocity.
- The mutation of the equations of motion from one stage to the next due to the varying of the exhaust velocity, v_e, or, possibly, other terms.
- The state discontinuities that occur at stage separation events from abrupt mass detachments.
- The fact that every trajectory optimization problem is a distinct boundary-value problem.

In generic terms, a dynamic system that is subject to non-differentiable points can be subdivided into different *phases* corresponding to periods of time along which the system is fully differentiable. This way, each phase is modelled with the appropriate set of dynamic equations and the non-differentiable points correspond to phase transition points. If there are multiple non-differentiable points then one can subdivide the problem into multiple phases. Always with the assumption that the transitions between phases are instantaneous.

Trajectory optimization problems of this sort are called *multiphase optimal control problems*.

In the particular case of multi-staged rockets, it is appropriate to subdivide the problems into phases according to the stage separation events. This way one stage is associated with one phase and stage separation events correspond to phase transition points. In the context of this work, the terms *stage* and *phase* might be used interchangeably.

The challenges described above are the motivation for the development of a multiphase numerical algorithm. There must be a trajectory generation utility capable of dealing with such systems. In addition, this utility would need to be generic instead of problem-specific in order to be useful for both current and future missions.

Path constraints

In addition to the difficulties expressed above, the vehicle might be subject to constraints due to design requirements or safety factors. For instance, it might be necessary to keep the aerodynamic pressure below a certain value along the trajectory, or it might be required to limit the acceleration experienced by the vehicle to an acceptable interval, or, still, the trajectory might be physically conditioned by no-fly zones. These types of constraints are called *path constraints* [6, 7] and they can be though of as an algebraic equation (or inequality) that must be verified at each point in time. Path constraints are expressed as

$$\boldsymbol{h}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \leq \mathbf{0}\,. \tag{1.13}$$

For reference, other examples of path constraints commonly used are:

- To assert a constant norm of the thrust vector along the trajectory.
- To ensure that the norm of the position vector is greater than or equal to the radius of the Earth.

Event constraints

Event constraints are a set of equations (or inequalities) that must be satisfied at the boundaries of the problem, that is, at the initial and/or final times of each phase. These are also referred to as *boundary conditions* or *point constraints* [7, 9, 10].

A commonly used set of event constraints in the case of OLVs is, for example, the assertion of orbital elements at the final time of the trajectory. When finding the optimal trajectory to a desired orbit, it is possible to assert the values of five out of the six orbital elements in order to leave one unknown, usually the true anomaly, as a degree of freedom. This is very convenient as, most often, the true anomaly of the spacecraft at the point of orbit insertion is not a relevant factor.

Event constraints are expressed as

$$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) \leq \mathbf{0}\,. \tag{1.14}$$

The equations of motion, together with the path constraints and the event constraints dictate feasibility. The state and control sequences that satisfy these constraints form a feasible solution, not necessarily an optimal solution.

The performance measure

The performance measure is a scalar quantity indicator of optimality [6–8, 11, 12]. In optimal control, this quantity is conventionally expressed as a *cost functional*, $\mathcal{J}$, this way the optimal trajectory is found by minimizing (or maximizing) the cost. The cost functional is expressed as [6–8, 11, 12]

$$\mathcal{J} = \overbrace{\underbrace{\Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big)}_{\text{Mayer Term}} + \int_{t_0}^{t_f} \underbrace{\Psi\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)}_{\text{Lagrange Term}} \, \mathrm{d}t}^{\text{Bolza Problem}} \, . \tag{1.15}$$

The Bolza problem is the standard formulation of a generic optimal control problem, standing as the sum of an integral functional of the continuous problem and a function evaluated at the boundaries [7, 11]. If the boundary term is omitted, $\Phi \equiv 0$, the problem is known as the *Lagrange problem*, and if the integrand term is omitted, $\Psi \equiv 0$, the problem is known as the *Mayer problem* [7, 11]. Most often it is possible to express a Lagrange problem as a Mayer problem (and vice-versa).

The performance measure is the parameter that dictates optimality: a trajectory that minimizes time will, most likely, be different to one that minimizes energy, yet both are optimal trajectories. Therefore, the process of finding the optimal trajectory begins by choosing the optimal performance measure. In the case of OLVs, the performance measure will often consist on maximizing the final mass of the vehicle.

1.3 Optimal Control and Trajectory Optimization Methods

Rocket trajectory optimization is profoundly tied to optimal control. Since launch vehicle technology developed (roughly during the second half of the twentieth century) the optimization of rocket trajectories has been made through Pontryagin's Minimum principle [5, 13]. The terms "trajectory optimization" and "optimal control" are so much tied together that they can often be used interchangeably [14]. In particular, the term *primer vector* was coined specifically in the context of rocket trajectory optimization [5, 13]. The primer vector is the complementary vector associated with velocity [5, 13, 15]. It has been shown that, for final-mass-maximization problems, one of the optimality conditions is for the thrust vector to be collinear with the primer vector [5, 13, 15]. The complementary vectors (or *covectors*, *costates*, *adjoined vectors*, or, still, *dual variables*) are auxiliary variables that serve a similar purpose to that of Lagrange multipliers in parametric optimization problems.

Given a constrained dynamic system, the subject of optimal control is concerned with finding the control sequence that will promote a desired state transformation and which will minimize (or maximize) an associated performance measure [6–8, 11, 12, 15]. Optimal control is a multidisciplinary subject that exists formally for more than three hundred years [16], although it is not unreasonable to assume that this discipline has been present in the collective human mind long before that in the shape of informal thoughts.

The classical approach to solve optimal control problems is through the calculus of variations, leading to the Hamiltonian boundary-value problem [6, 8]. Most often analytical solutions cannot be derived, and thus numerical methods are employed. Until the beginning of the twenty-first century indirect methods, such as

single shooting and multiple shooting methods, were standard for solving trajectory optimization problems, including rocket and spacecraft trajectories [6, 8, 13, 17, 18].

But indirect methods are often impractical due to the associated necessity of deriving the problem-wise optimality conditions [7]. In addition, these methods most often comprise convergence issues [7, 19], and thus, lack reliability when a good initial guess is not available. On top of this, indirect methods are inadequate for problems with inequality path constraints: typically the time intervals relative to the active and the inactive constraints must be known a priori so that the problem can be manually divided into corresponding phases [7]. Ultimately, these issues make indirect methods hostile to employ in a general purpose software application.

In contrast, direct methods, and in particular pseudospectral methods, have gained popularity over the past decades since their introduction by Elnagar *et al.* [20] and their generalization by Ross, Fahroo *et al.* [21–24]. Pseudospectral methods consist on the transcription of the dynamics based on the roots of an orthogonal polynomial, typically a particular Jacobi polynomial [9, 25]. The constraints are evaluated at each collocation point resulting in a large list of equality and inequality algebraic constraints, accompanied with the algebraic objective function; this list of constraints, referred to as a nonlinear programming (NLP) problem, is passed through an *off-the-shelf* nonlinear optimizer such as SNOPT [26] or IPOPT [27, 28] and the solution is obtained. In essence, in a pseudospectral method the optimal control problem is transformed into a parametric optimization problem.

The solution obtained from a pseudospectral method is computed with disregards to the first order optimality conditions from the Hamiltonian Boundary-Value Problem. Instead, the set of necessary conditions associated with the NLP is employed, called the Karush–Kuhn–Tucker conditions [19], resulting in some loss of information with regards to the original optimal control problem [23, 29]. In order to make sure that the solution is optimal, the first order necessary conditions are verified in post processing with the aid of the covector mapping theorem (CMT) [23, 29–31], thus "double checking" the optimality of the solution. Finally, having the discrete solution, the continuous representation can be obtained by use of Lagrange interpolating polynomials [32]. Pseudospectral methods can also be referred to as *orthogonal collocation methods* and they are known to converge spectrally [23, 33, 34].

Since the introduction of pseudospectral methods there have been multiple iterations and innovations [19, 29, 35–37], including their implementation into several software tools such as GPOPS and GPOPS-II [9, 10], DIDO [21, 38], PSOPT [39], ICLOCS2 [40], SPARTAN [32, 41–46] among others. One notable instance of the application of pseudospectral methods into the real world was the famous *zero-propellant manoeuvre* of the ISS where a large angle reorientation of the spacecraft was performed by aid of reaction wheels and gravity gradient only, virtually no propellant was spent [47]. Direct methods are widely used because they can accommodate a broader range of problem formulations than indirect methods and lack their drawbacks.

The NLP problem passed to the nonlinear solver via a pseudospectral method requires the aid of an associated Jacobian matrix. Because most optimal control software packages are general purpose, they must either (i) require the user for an analytic Jacobian matrix which can quickly become an exhausting process, especially for larger and more complicated problems, or (ii) the software packages provide the Jacobian matrix via an arbitrary numerical method. For convenience to the user, the second alternative is most often provided.

Because of this, these software packages require numerical differentiation methods to compute the partial derivatives of the constraints with respect to the variables and ultimately assemble the Jacobian matrix. In the context of optimal control, methods such as the *complex step* method and the *Dual number* method have been employed to some extent [48–52].

Because generality is desired, a scaling process is often applied to the NLP as a way to normalize the magnitude of the algebraic constraints. One effective scaling method is to factor each constraint by the magnitude of the respective Jacobian matrix row [51, 53, 54]. Another scaling method has been proposed recently which is based on a balancing technique between primal and dual variables [55]. Despite not introducing coupling between the constraints, the scaling does affect the resulting output dual variables. This means that the NLP needs to be rescaled back in post-processing before the output of the final solution and also before the covector mapping.

Within pseudospectral methods, the three most commonly used techniques are the Legendre-Gauss pseudospectral method, the Legendre-Gauss-Lobatto pseudospectral method and the Legendre-Gauss--Radau pseudospectral method (direct or flipped), all of which have been properly formalized [29]. The roots of Legendre polynomials sit invariably within the interval $[-1, +1]$. The position of the roots changes slightly from method to method, but the most relevant difference between the methods is the inclusion (or exclusion) of the interval endpoints (-1) or (+1). The roots of the Legendre-Gauss polynomial do not include any of the endpoints; the roots of the Legendre-Gauss-Lobatto polynomial include both interval endpoints; and the roots of the Legendre-Gauss-Radau polynomial include either the left endpoint (-1) or the right endpoint (+1) of the interval.

The roots of the Legendre-Gauss polynomial do not include the endpoints of the interval, because of this, the respective method by itself does not allow for endpoint constraints, in order to include endpoint constraints, one must add an auxiliary quadrature constraint for artificial collocation of the terminal (or initial) endpoint [9]. This extra constraint constitutes an additional burden to the NLP, making the problem slightly, but unnecessarily, more complex. The Legendre-Gauss-Lobatto method provides a solution to this because, without the necessity of additional constraints, the method allows both endpoints of the domain to be constrained. However, it has been shown that this method has convergence issues with respect to the costates [29, 37] which compromises the calculation of the first order optimality conditions. Finally, the Legendre-Gauss-Radau method (direct or flipped), due to the asymmetric distribution of collocation points, encompasses the best of both worlds by allowing the domain endpoints to be constrained without sacrificing convergence of the costates [37].

With regards to pseudospectral methods, there have been efforts to develop the so-called *integral formulation* of the NLP as an alternative to the typical differential form [10, 35, 56, 57]. It has been shown that the choice of the differential form over the integral form or vice-versa is arbitrary for both the Legendre-Gauss pseudospectral method and the Legendre-Gauss-Radau pseudospectral method (direct or flipped). However, this is not the case for the Legendre-Gauss-Lobatto pseudospectral method due to the structure of the respective differentiation matrix.

Regarding multiphase optimal control, successful efforts have been made in order to implement the feature into software [9, 10]. Multiphase optimal control allows the study of trajectories that might contain

necessary state discontinuities or even mutations of the equations of motion. Figure 1.2 illustrates the launch sequence of a vehicle with two stages. It is evident that the stage separation event introduces a discontinuity of mass into the system. In order to generate trajectories of orbital launchers it is relevant to develop a multiphase algorithm. This way state discontinuities can be accounted for.

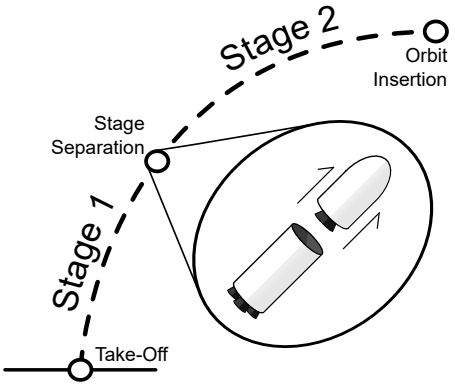


Figure 1.2: Simplified launch sequence illustration of a two staged rocked.

Other developments in pseudospectral optimal control have taken place, most notably *h* and *hp* mesh refinement methods [52, 58–60]. In a *h* method the number of polynomials to be concatenated along the domain is increased, while in a *p* method the degree of the polynomial in use is increased. *hp* methods are a combination of the two. The implementation of *p* mesh refinement methods is often superfluous, as it is more satisfying to manually modify the degree of the polynomial at rerun whenever a solution is not considered accurate enough. On the other hand *h* and *hp* methods can be powerful, however, the respective implementation can be laborious and a refinement strategy is required. Other mesh refinement methods have been proposed which are based on discontinuities in the optimal control [61–63], this is relevant because the control discontinuity points are ideal to concatenate adjacent mesh intervals.

In this work, the flipped Radau pseudospectral method is employed to generate optimal trajectories of multi-stage rockets in both ascension to orbit and descent to vertical landing. The Legendre-Gauss-Radau method is well suited because it is a pseudospectral method that allows endpoint collocation, and it does not present dramatic convergence issues on the dual variables [37]. The foundation for the implementation of the multiphase feature is SPARTAN [32], a MATLAB tool developed at DLR — Deutsches Zentrum für Luft- und Raumfahrt — with the objective of solving general purpose optimal control problems.

Chapter 2

Direct and Indirect Method Paths and Covector Mapping

2.1 The Optimal Control Problem

An optimal control problem (OCP) is a formal mathematical statement describing a physical scenario where a minimization objective is required and a system of ordinary differential equations (ODEs) is involved. One characteristic of optimal control, in contrast to nonlinear programming (pure parametric optimization), is that the system in study is subject to dynamic constraints as well as algebraic ones [6, 7]. This means that the passing of time, or the domain of a relevant independent variable, is involved in some way. The problems become more complicated and interesting. An optimal control problem can be formulated as [6, 8, 37] minimizing the cost functional

$$\mathcal{J} = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) + \int_{t_0}^{t_f} \Psi\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)\,\mathrm{d}t\ , \tag{2.1}$$

associated with the trajectory of the dynamic system

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)\ , \tag{2.2}$$

subject to the path constraints

$$\boldsymbol{h}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \leq \boldsymbol{0}\,, \tag{2.3}$$

and to the event constraints

$$\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) \leq \boldsymbol{0}\,. \tag{2.4}$$

The functions above are defined by the following mappings:

$$\Phi : \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}\,, \tag{2.5}$$

$$\Psi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}\,, \tag{2.6}$$

$$\boldsymbol{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}\,, \tag{2.7}$$

$$\boldsymbol{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \rightarrow \mathbb{R}^{n_h}, \tag{2.8}$$

$$\boldsymbol{\phi} : \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \rightarrow \mathbb{R}^{n_\phi}, \tag{2.9}$$

where $n_{\boldsymbol{h}}$ is the dimension of the path constraint vector, i.e., the number of simultaneous constraints that the system is subject to at a given time and $n_{\boldsymbol{\phi}}$ is the dimension of the event constraint vector.

An optimal control problem exists in the domain of an independent variable about which integration occurs. In the context of geometric problems, for instance, the independent variable can be defined as a spatial coordinate, while in the case of dynamic systems such as orbital launchers, the independent variable is time.

2.2 Mathematical Background

Solving optimal control problems numerically requires discretization, regardless of the method in use. The discretized problem must be managed similarly to the continuous one, therefore it requires equivalent mathematical operators. For example, there must be a discrete integral operator, as well as a discrete derivative operator.

In this Section some relevant mathematical concepts are briefly reviewed, namely the *Gaussian quadrature* which is a discrete integral operator, and the differentiation matrix based on Lagrange polynomial interpolation used to compute a discrete derivative. Other relevant concepts are presented, namely the linear mapping of the domain of an optimal control problem, as well as some properties of the Legendre-Radau polynomials.

2.2.1 Gaussian Quadrature and Domain Mapping

It is possible to compute the definite integral from –1 to 1 of a given polynomial function, $f(\tau)$, according to the *Gaussian quadrature*. The function is evaluated at N discrete points and a weighted sum of the samples is performed [29, 64, 65],

$$\int_{-1}^{1} f(\tau)\, \mathrm{d}\tau = \sum_{k=1}^{N} w_k f_k \,, \tag{2.10}$$

where $f_k = f(\tau_k)$ is the value of the function at $\tau = \tau_k$, and w_k is a scalar quadrature weight associated with the k^{th} function sample. In the case of a function that is constant, for instance $f(\tau) = 1$, the quadrature rule is [65]

$$\int_{-1}^{1} \mathrm{d}\tau = \sum_{k=1}^{N} w_k = 2 \,. \tag{2.11}$$

It is important to note that it is not sufficient to evaluate the function at any given set of N discrete points. For the quadrature to be accurate it is crucial that the distribution of points follows a pattern similar to the distribution of roots of a Legendre-based polynomial (also called Gauss points) [65]. For example, by sampling the function according to the roots of a N^{th} order Legendre-Radau polynomial the Gaussian quadrature is exactly accurate for polynomial integrand functions of order up to $2N - 2$ [29, 64]. If the integrand function is not a polynomial then there will be an error associated with the integration [29, 65]. This error can be reduced

by increasing the number of sample points, N, effectively approximating the integrand by a polynomial of higher order.

In order to perform an integration over a generic interval $t \in [t_0, t_f]$, the domain of the independent variable, t, is mapped into the normalized domain $\tau \in [-1, 1]$. This mapping is linear and it can be expressed as [51]

$$t(\tau) = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2}, \quad \tau \in [-1, 1]\,, \tag{2.12}$$

$$\mathrm{d}t = \frac{t_f - t_0}{2}\,\mathrm{d}\tau\ . \tag{2.13}$$

By performing a change of variables, the Gaussian quadrature can be expressed for a generic interval as

$$\int_{t_0}^{t_f} f(t)\,\mathrm{d}t = \frac{t_f - t_0}{2}\sum_{k=1}^{N} w_k f_k\,, \tag{2.14}$$

where $f_k = f\big(t(\tau_k)\big)$ is the value of the function evaluated at $t(\tau_k)$.

The change of variable of integration can be interpreted as a shift in the domain from concrete time, t, to a *pseudo-time*, τ, which is always within –1 to 1. This domain shift allows for the problem to be normalized and for the endpoints of the domain, t_0 and t_f, to be set as optimization parameters (decision variables), which is crucial when the final time of the trajectory is unknown, for instance.

2.2.2 Lagrange Polynomial Interpolation and the Differentiation Matrix

With a set of N discretization points along the domain, and the corresponding samples of a polynomial $\boldsymbol{x}_i$, the smooth function approximation, $\boldsymbol{x}(\tau)$, which passes through the N points can be found via *Lagrange polynomial interpolation* by [66]

$$\boldsymbol{x}(\tau) = \sum_i \boldsymbol{x}_i L_i(\tau)\,, \tag{2.15}$$

$$L_i(\tau) = \prod_{\substack{j \\ j\neq i}} \frac{\tau - \tau_j}{\tau_i - \tau_j}\,, \tag{2.16}$$

where τ_i is the independent variable associated with the i^{th} sample point and $L_i(\tau)$ is the corresponding Lagrange interpolating polynomial which is a function of the independent variable τ and a function of the locations of the samples along the domain.

The first derivative of the polynomial, $\boldsymbol{x}(\tau)$, with respect to τ can also be estimated based on the Lagrange polynomial interpolation by [66]

$$\frac{\mathrm{d}\boldsymbol{x}(\tau)}{\mathrm{d}\tau} = \sum_i \boldsymbol{x}_i \frac{\mathrm{d}L_i(\tau)}{\mathrm{d}\tau}\,. \tag{2.17}$$

Alternatively, a differentiation matrix, $\boldsymbol{D}$, can be built such that the derivative of $\boldsymbol{x}$ with respect to τ at the points $\tau = \tau_k$ is computed as

$$\frac{\mathrm{d}\boldsymbol{x}_k}{\mathrm{d}\tau} = \sum_i \boldsymbol{D}_{ki}\boldsymbol{x}_i\,, \tag{2.18}$$

where $\boldsymbol{x}_k = \boldsymbol{x}(\tau_k)$ and $\boldsymbol{D}_{ki}$ is a scalar element on the k^{th} row and i^{th} column of matrix $\boldsymbol{D}$. This differentiation

matrix can be constructed one element at a time as [9]

$$\boldsymbol{D}_{ki} = \frac{\mathrm{d}L_i(\tau_k)}{\mathrm{d}\tau} = \sum_{\substack{l \\ l \neq i}} \frac{1}{\tau_i - \tau_l} \prod_{\substack{j \\ j \neq i,l}} \frac{\tau_k - \tau_j}{\tau_i - \tau_j} \,. \tag{2.19}$$

Applying the chain rule to (2.18) and using (2.13), the derivative of $\boldsymbol{x}$ with respect to a generic domain, $t \in [t_0, t_f]$, can be computed by [23]

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\mathrm{d}\tau}{\mathrm{d}t}\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\tau} \,, \tag{2.20}$$

$$\dot{\boldsymbol{x}}_k = \frac{\mathrm{d}\boldsymbol{x}_k}{\mathrm{d}t} = \frac{2}{t_f - t_0} \sum_i \boldsymbol{D}_{ki}\boldsymbol{x}_i \,. \tag{2.21}$$

As was the case for Gaussian quadrature, the accuracy of the Lagrange polynomial interpolation, and that of the differentiation matrix, is directly related to the choice of sample points along the domain of τ. Equation (2.21) provides a generic way to compute the first derivative of any given function, $\boldsymbol{x}(t)$, with respect to a generic domain, provided an appropriate sampling strategy is used. If no mind is paid to the selection of discretization points, for instance, by choosing a uniform distribution of nodes, the accuracy of the Lagrange approximation is not guaranteed, mainly due to the Runge phenomenon. As it happens, the roots of the three Legendre-based polynomials mentioned earlier (Legendre, Legendre-Radau and Legendre-Lobatto) turn out to be a good choice for the discretization pattern [29].

Although the derivative $\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}$ could be computed directly by building matrix $\boldsymbol{D}$ using the values of t instead of τ in (2.16) and (2.19), it is more convenient to isolate t_0 and t_f as these will be used as decision variables. And so, matrix $\boldsymbol{D}$ is used for the derivative with respect to the normalized domain of τ and the derivative with respect to t yields easily via a simple multiplication. This also has the advantage of matrix $\boldsymbol{D}$ being constant, so it will not change as t_0 or t_f change.

2.2.3 Legendre-Radau polynomial and node distribution

Due to the Runge phenomenon, the worst possible choice of sample points for polynomial interpolation is an equidistant grid [29, 51]. In contrast, a good choice of sample points is a proportional mapping of the roots of Legendre-based polynomials, such as the *flipped Radau polynomial*. The flipped Radau Polynomial is the result of the difference between two Legendre polynomials of consecutive order:

$$R_N(\tau) = P_N(\tau) - P_{N-1}(\tau), \quad \tau \in [-1, 1] \tag{2.22}$$

where $P_N(\tau)$ is the Legendre polynomial of order N. Figure 2.1 shows the comparison between an equidistant grid and the distribution of the flipped Radau polynomial roots for reference. One peculiarity of the roots of the Radau polynomial is that they are asymmetric with respect to the origin.

The Gaussian quadrature weights associated with the roots of the flipped Radau polynomial can be computed by doing a *flip* operation on the weights of the direct Radau, reversing their order [29, 51]

$$w_k = \mathtt{flip}\left\{ \frac{1 - \tau_k^{\mathrm{DR}}}{N^2 P_{N-1}^2(\tau_k^{\mathrm{DR}})} \right\}, \quad k = 1, 2, .., N \,, \tag{2.23}$$

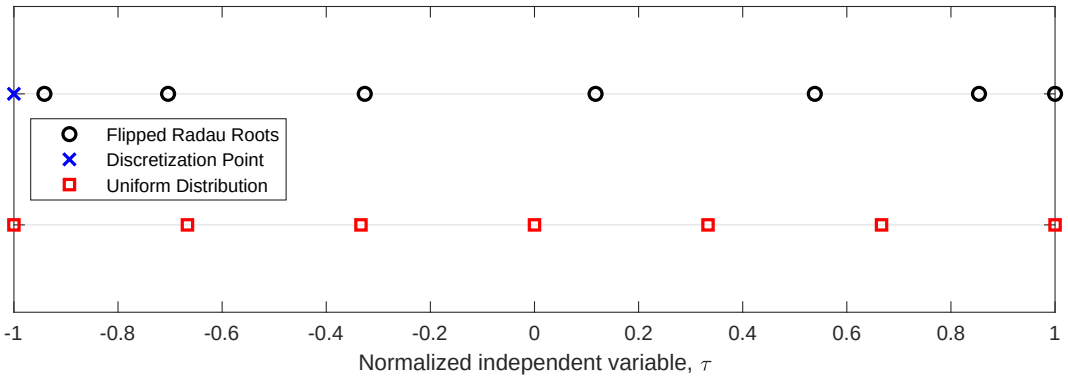


Figure 2.1: Comparison of node distributions for $N = 7$ in the domain $\tau \in [-1, 1]$. A uniform distribution is shown in red squares. The roots of the flipped Radau polynomial are shown in black circles, an extra discretization point at $\tau = -1$ is shown as a blue cross.

where τ_k^{DR} is the k^{th} abscissa of the direct Radau roots, N is the degree of the Radau polynomial, and $P_{N-1}(\tau_k^{\mathrm{DR}})$ is the Legendre polynomial of degree $N-1$ evaluated at τ_k^{DR}.

Because the roots of the flipped Radau polynomial do not include the left limit of the interval ($\tau = -1$) there will be no weight associated with this point, and therefore this point will not be collocated. However the point at $\tau = -1$ can be used as an auxiliary discretization sample for the calculation of the derivative at the collocation points [35]. Also, this additional discretization point allows the specification of the state at the beginning of the trajectory (initial condition).

Letting $i = 0, 1, 2, \ldots, N$ be the index corresponding to the discretization points, and $k = 1, 2, \ldots, N$ be the index corresponding to the collocation points, the resulting differentiation matrix $\boldsymbol{D}$ from (2.19) will be rectangular of size $N \times (N+1)$.

2.3 Indirect Method Route

In this Section, the classical indirect method approach to solving optimal control problems is reviewed. In an indirect method the optimal control problem is transcribed by way of the following two steps:

1. Dualization,
2. Discretization,

(in that order) after which a solver procedure is employed to obtain a discretized solution.

The two steps described above are further explored in this Section in the following way: Section 2.3.1 describes step 1, presenting the dualization process and the optimality conditions known as the *Hamiltonian Boundary-Value Problem*, and Section 2.3.2 briefly describes step 2, where only the discrete version of the cost functional is presented.

2.3.1 Hamiltonian Boundary-Value Problem

In an indirect method the process of finding a solution to the optimal control problem begins by deriving the first order optimality conditions. In order to do so, the cost functional (2.1) is augmented by means of the complementary vectors, $\boldsymbol{\nu}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, (Lagrange multipliers) to include all constraints expressed in (2.2) to

(2.4). This process is often referred to as *dualization* [19]. The augmented cost functional is written as [37]

$$\begin{aligned}\mathcal{J}^{\lambda} = \Phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) + \boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) \\ + \int_{t_0}^{t_f} \Big\{\Psi\big(\boldsymbol{x}, \boldsymbol{u}\big) + \boldsymbol{\lambda}^{\mathsf{T}}\big[\boldsymbol{f}\big(\boldsymbol{x}, \boldsymbol{u}\big) - \dot{\boldsymbol{x}}\big] + \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{h}\big(\boldsymbol{x}, \boldsymbol{u}\big)\Big\}\,\mathrm{d}t\ ,\end{aligned} \tag{2.24}$$

where $\boldsymbol{\lambda}(t) \in \mathbb{R}^{n_x}$ is the state covector (or costate), $\boldsymbol{\mu}(t) \in \mathbb{R}^{n_h}$ is the constraint covector and $\boldsymbol{\nu} \in \mathbb{R}^{n_\phi}$ is the endpoint covector. Two important quantities that simplify the writing of the first order optimality conditions are the variational Hamiltonian, $\mathcal{H}$, and the Lagrangian of the Hamiltonian, $\mathcal{L}$, which can be expressed as [23]:

$$\mathcal{H}\big(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}\big) = \Psi\big(\boldsymbol{x}, \boldsymbol{u}\big) + \boldsymbol{\lambda}^{\mathsf{T}}\boldsymbol{f}\big(\boldsymbol{x}, \boldsymbol{u}\big)\,, \tag{2.25}$$

$$\mathcal{L}\big(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}\big) = \Psi\big(\boldsymbol{x}, \boldsymbol{u}\big) + \boldsymbol{\lambda}^{\mathsf{T}}\boldsymbol{f}\big(\boldsymbol{x}, \boldsymbol{u}\big) + \boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{h}\big(\boldsymbol{x}, \boldsymbol{u}\big)\,. \tag{2.26}$$

In order to introduce the concept of variations Fig. 2.2 illustrates two hypothetical trajectories: an optimal trajectory, $\boldsymbol{x}^*(t)$, and a neighbouring solution $\boldsymbol{x}(t)$. The image presents a simplified version of the relationships between these two functions and the variations at the endpoints, namely δt_0, δt_f, $\delta \boldsymbol{x}_0$ and $\delta \boldsymbol{x}_f$. It can be seen that if t_f and $\boldsymbol{x}(t_f)$ are not allowed to vary, then the optimal solution must invariably include the right endpoint.

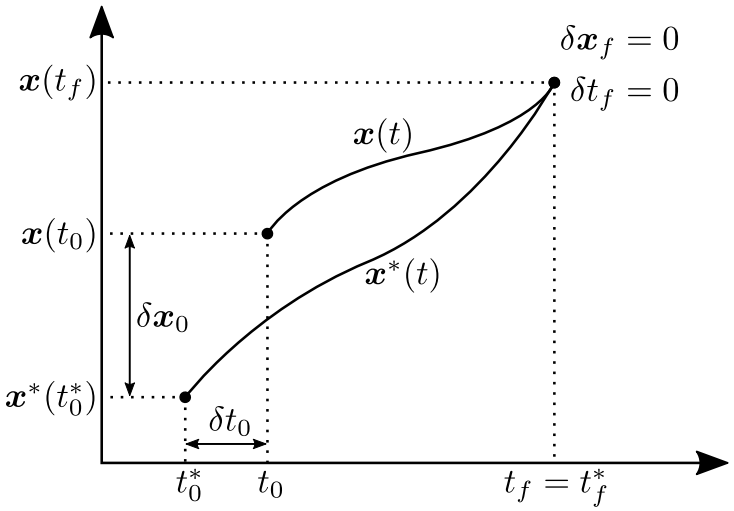


Figure 2.2: Example trajectories for illustration of the variations δt_0, δt_f, $\delta \boldsymbol{x}_0$ and $\delta \boldsymbol{x}_f$. An optimal trajectory is represented by $\boldsymbol{x}^*(t)$ and a varying neighbour solution is represented by $\boldsymbol{x}(t)$.

The first order necessary conditions for optimality can be derived from (2.24) by setting the first variation of the augmented cost functional equal to zero, $\delta \mathcal{J}^{\lambda} = 0$. These necessary conditions are expressed in terms of the Lagrangian of the Hamiltonian as [6, 8, 23, 29]:

$$\boldsymbol{f}\big(\boldsymbol{x}, \boldsymbol{u}\big) - \dot{\boldsymbol{x}} = \boldsymbol{0}\,, \tag{2.27}$$

$$\boldsymbol{\mu}^{\mathsf{T}}\boldsymbol{h}\big(\boldsymbol{x}, \boldsymbol{u}\big) = 0\,, \tag{2.28}$$

$$\boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\phi}\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) = 0\,, \tag{2.29}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}}\big(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}\big) = \boldsymbol{0}\,, \tag{2.30}$$

$$\dot{\boldsymbol{\lambda}}^{\mathsf{T}} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}\big(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}\big) = \boldsymbol{0}\,, \tag{2.31}$$

$$\delta \boldsymbol{x}_0 = \mathbf{0} \quad \veebar \quad \frac{\partial \Phi}{\partial \boldsymbol{x}(t_0)} + \boldsymbol{\nu}^\intercal \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_0)} + \boldsymbol{\lambda}^\intercal(t_0) = \mathbf{0}\,, \tag{2.32}$$

$$\delta \boldsymbol{x}_f = \mathbf{0} \quad \veebar \quad \frac{\partial \Phi}{\partial \boldsymbol{x}(t_f)} + \boldsymbol{\nu}^\intercal \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}(t_f)} - \boldsymbol{\lambda}^\intercal(t_f) = \mathbf{0}\,, \tag{2.33}$$

$$\delta t_0 = 0 \quad \veebar \quad -\mathcal{H}\big(t_0\big) + \frac{\partial \Phi}{\partial t_0} + \boldsymbol{\nu}^\intercal \frac{\partial \boldsymbol{\phi}}{\partial t_0} = 0\,, \tag{2.34}$$

$$\delta t_f = 0 \quad \veebar \quad \mathcal{H}\big(t_f\big) + \frac{\partial \Phi}{\partial t_f} + \boldsymbol{\nu}^\intercal \frac{\partial \boldsymbol{\phi}}{\partial t_f} = 0\,, \tag{2.35}$$

where $\mathcal{H}\big(t_0\big)$ and $\mathcal{H}\big(t_f\big)$ is the Hamiltonian evaluated at t_0 and t_f, respectively:

$$\mathcal{H}\big(t_0\big) = \mathcal{H}\big(\boldsymbol{x}(t_0), \boldsymbol{u}(t_0), \boldsymbol{\lambda}(t_0)\big)\,, \tag{2.36}$$

$$\mathcal{H}\big(t_f\big) = \mathcal{H}\big(\boldsymbol{x}(t_f), \boldsymbol{u}(t_f), \boldsymbol{\lambda}(t_f)\big)\,. \tag{2.37}$$

Equations (2.27), (2.28) and (2.29) result from the variation of the augmented cost about the three covectors, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, respectively, and they are a rewriting of the constraints of the original problem statement in (2.2) through (2.4). One interesting result that yields from the optimality conditions is the fact that the inequality constraints (2.3) and (2.4) are turned into equality constraints by means of a product with the respective covectors: the conditions are satisfied whenever the inequality constraints are active ($\boldsymbol{h} = \mathbf{0}$ and/or $\boldsymbol{\phi} = \mathbf{0}$), and they can be satisfied when the constraints are inactive ($\boldsymbol{h} < \mathbf{0}$ and/or $\boldsymbol{\phi} < \mathbf{0}$) by making sure that the covectors bind to zero at the corresponding times ($\boldsymbol{\mu} = \mathbf{0}$ and/or $\boldsymbol{\nu} = \mathbf{0}$), thus ensuring a null product in (2.28) and (2.29). Because of this, (2.28) and (2.29) are often referred to as *slackness conditions*.

Equations (2.30) and (2.31) are called the *Euler-Lagrange* equations and they apply at every point along the domain. In particular (2.31) is called the *costate equation* because it describes the rate of change of the costate, effectively serving as a new equation of motion corresponding to the costates. Equation (2.30) is also referred to as the strong form of Pontryagin's minimum principle and it only applies if the optimal control sequence lies exclusively within the allowable control set. Thus, equation (2.30) must be disregarded in the case of "bang-bang" control problems where the control sequence includes a saturation of the control set.

Finally (2.32) through (2.35) are the endpoint conditions and they are characterized by an exclusive disjunction property, represented by the symbol "$\veebar$". Briefly stated, the right-hand-side conditions only apply when the left-hand-side conditions do not, and vice-versa. For instance, if in a given problem the initial time is known (fixed), then its variation is zero $\delta t_0 = 0$, this means that the left-hand-side of (2.34) applies, but the right-hand-side does not. The right-hand-side equation of (2.34) only applies when the variation of the initial time is not zero, i.e, when the initial time is not known. The analogous is true for the other endpoint conditions. The exclusive disjunction property of the endpoint conditions results in an incomplete set of conditions in either endpoint of the domain, which is why an optimal control problem is essentially a *boundary-value problem*.

In addition, because it is not an explicit function of time, the Hamiltonian of the optimal solution will be a constant [37]. Thus:

$$\mathcal{H}\big(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\lambda}(t)\big) = \mathcal{H}^\dagger\,, \tag{2.38}$$

$$\mathcal{H}(t_0) = \mathcal{H}(t_f) = \mathcal{H}^\dagger\,, \tag{2.39}$$

where $\mathcal{H}^\dagger$ represents a generic constant scalar.

The problem of finding a solution that satisfies all first order optimality conditions is known as the *Hamiltonian boundary value problem* (HBVP). If a problem is simple enough, then it might be possible to solve it analytically, however, due to the nonlinear nature of optimal control problems, most often an analytical solution is not possible to obtain and numerical methods must be employed. Therefore, the HBVP has to be discretized.

2.3.2 Discrete Hamiltonian Boundary-Value Problem

The *discrete Hamiltonian boundary-value problem* results form the discretization of the first order optimality conditions discussed in Section 2.3.1. This discretization is necessary in order to make the problem solvable by numerical methods. For the sake of argument, let the optimality conditions be discretized according to the roots of the flipped Radau polynomial. Then, from (2.25) and (2.26), the discrete Hamiltonian and Lagrangian are:

$$\mathcal{H}_k = \mathcal{H}\big(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\lambda}_k\big) \qquad = \Psi_k + \boldsymbol{\lambda}_k^\intercal \boldsymbol{f}_k \,, \tag{2.40}$$

$$\mathcal{L}_k = \mathcal{L}\big(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k\big) = \Psi_k + \boldsymbol{\lambda}_k^\intercal \boldsymbol{f}_k + \boldsymbol{\mu}_k^\intercal \boldsymbol{h}_k \,, \tag{2.41}$$

and the augmented-discrete cost functional can be can be written based on the discrete optimality conditions as

$$\mathcal{J}^{\lambda N} = \Phi + \boldsymbol{\nu}^\intercal \boldsymbol{\phi} + \frac{t_f - t_0}{2} \sum_k w_k \left[\mathcal{L}_k - \boldsymbol{\lambda}_k^\intercal \frac{2}{t_f - t_0} \sum_i \boldsymbol{D}_{ki} \boldsymbol{x}_i \right] . \tag{2.42}$$

Another way to obtain (2.42) is to discretize (2.24) directly, this way the integration becomes a Gaussian quadrature and the time derivative, $\dot{\boldsymbol{x}}$, becomes a multiplication with the differentiation matrix. Again, for this discretization to be meaningful the distribution of nodes in the domain of time must be similar to the root pattern of the flipped Radau polynomial or of some other Legendre based polynomial. For brevity, the discrete optimality conditions derived from the HBVP are not presented.

Indirect methods have been the standard for solving optimal control problems because they are intuitive and allow for an analytical approach, also, there is no loss of information in the process which means that once a solution is obtained, there is no need to double check its validity. However, these methods require the derivation of the problem-wise optimality conditions, which means that they are unsuited for a general purpose software program. Indirect methods have been the standard to solve optimal control problems until the beginning of the twenty-first century, then, by the ever-increasing computing power of modern machines, direct methods became a reliable alternative.

2.4 Direct Method Route

In this Section the direct method approach to solve optimal control problems is reviewed. The transcription process of a direct method is composed of the same two steps as the indirect method approach described in Section 2.3, only *the steps are in reverse order*:

1. Discretization
2. Dualization

In this Section these two steps are reviewed in the context of direct methods in the following manner: in Section 2.4.1 describes the discretization step, which transforms the original optimal control problem into a nonlinear programming problem, and in Section 2.4.2 the optimality conditions of the NLP, known as the *Karush-Kuhn-Tucker conditions*, are presented.

2.4.1 Nonlinear Programming Problem (NLP)

In a direct method, the optimal control problem, stated in (2.1) through (2.4), is first discretized. The result is a list of discrete algebraic constraints which is commonly referred to as a nonlinear programming problem (NLP). The NLP is sent to a nonlinear solver such as such as SNOPT [26] or IPOPT [27, 28] which iterates on the optimization variables until a solution is obtained. The NLP can be expressed as minimizing the discrete cost

$$\mathcal{J}^N = \Phi + \frac{t_f - t_0}{2} \sum_k w_k \Psi_k \,, \tag{2.43}$$

subject to:

$$C_{\lambda,k} \left[\boldsymbol{f}_k - \frac{2}{t_f - t_0} \sum_i \boldsymbol{D}_{ki} \boldsymbol{x}_i \right] = \boldsymbol{0} \,, \tag{2.44}$$

$$C_{\mu,k} \boldsymbol{h}_k \leq \boldsymbol{0} \,, \tag{2.45}$$

$$\boldsymbol{\phi} \leq \boldsymbol{0} \,, \tag{2.46}$$

where (2.43) results from substituting the integral term in (2.1) with (2.14), and (2.44) yields by expanding the term $\dot{\boldsymbol{x}}_k$ in $\boldsymbol{f}_k - \dot{\boldsymbol{x}}_k = \boldsymbol{0}$ with (2.21). The scalar factors $C_{\lambda,k}$ and $C_{\mu,k}$ are introduced in (2.44) and (2.45), respectively, in order to facilitate the generalization of the *covector mapping theorem*. As long as the scalars are not zero, they do not affect the solution of the primal variables (state and control), but they do affect the solution of the dual variables, and this can be used to obtain a meaningful mapping of the covectors. The equality constraints imposed by (2.44) (or other similar forms) will be referred to as "dynamic defects" throughout this work.

2.4.2 Karush–Kuhn–Tucker Conditions

The *Karush–Kuhn–Tucker (KKT) conditions* are the first order optimality conditions that result from the dualization of the NLP. Thus, the KKT conditions are to the NLP what the Hamiltonian Boundary-Value Problem (Section 2.3.1) is to the original optimal control problem.

In order to express the KKT conditions, two new quantities are introduced which relate to (2.25) and (2.26), these are the hereinafter called KKT Hamiltonian and the KKT Lagrangian, respectively:

$$\widetilde{\mathcal{H}}_k = \widetilde{\mathcal{H}}\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \tilde{\boldsymbol{\lambda}}_k, t_0, t_f\right) \quad = \frac{t_f - t_0}{2} w_k \Psi_k + C_{\lambda,k} \tilde{\boldsymbol{\lambda}}_k^\intercal \boldsymbol{f}_k \,, \tag{2.47}$$

$$\widetilde{\mathcal{L}}_k = \widetilde{\mathcal{L}}\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \tilde{\boldsymbol{\lambda}}_k, \tilde{\boldsymbol{\mu}}_k, t_0, t_f\right) = \frac{t_f - t_0}{2} w_k \Psi_k + C_{\lambda,k} \tilde{\boldsymbol{\lambda}}_k^\intercal \boldsymbol{f}_k + C_{\mu,k} \tilde{\boldsymbol{\mu}}_k^\intercal \boldsymbol{h}_k \,. \tag{2.48}$$

The discrete-augmented cost functional is expressed in terms of the KKT Lagrangian as [23, 29]

$$\mathcal{J}^{N\lambda} = \Phi + \tilde{\boldsymbol{\nu}}^{\intercal}\boldsymbol{\phi} + \sum_{k}\left[\widetilde{\mathcal{L}}_k - \tilde{\boldsymbol{\lambda}}_k^{\intercal} C_{\lambda,k}\frac{2}{t_f - t_0}\sum_i \boldsymbol{D}_{ki}\boldsymbol{x}_i\right]. \tag{2.49}$$

The dualization of the NLP occurs within the nonlinear solver, the KKT multipliers, $\tilde{\boldsymbol{\lambda}}_k$, $\tilde{\boldsymbol{\mu}}_k$ and $\tilde{\boldsymbol{\nu}}$, are attributed automatically and thus the KKT conditions are "invisible" to the outside, being contained in the "black box" of the solver. This means that the user only needs to worry about the NLP and is never required to derive optimality conditions, regardless of the problem at hand. This is a great advantage of direct methods because the implementation of these methods into general purpose software is far easier than that of indirect methods. The KKT conditions can be expressed in accordance with the current nomenclature as [23, 29]:

$$C_{\lambda,k}\left[\boldsymbol{f}_k - \frac{2}{t_f - t_0}\sum_i \boldsymbol{D}_{ki}\boldsymbol{x}_i\right] = \boldsymbol{0}\,, \tag{2.50}$$

$$\tilde{\boldsymbol{\mu}}_k^{\intercal} C_{\mu,k}\boldsymbol{h}_k = 0\,, \tag{2.51}$$

$$\tilde{\boldsymbol{\nu}}^{\intercal}\boldsymbol{\phi} = 0\,, \tag{2.52}$$

$$\frac{\partial\widetilde{\mathcal{L}}_k}{\partial\boldsymbol{u}_k} = \boldsymbol{0}\,, \tag{2.53}$$

$$w_k\sum_i \frac{2C_{\lambda,i}}{(t_f - t_0)w_i}\boldsymbol{D}^*_{ki}\tilde{\boldsymbol{\lambda}}_i^{\intercal} + \frac{\partial\widetilde{\mathcal{L}}_k}{\partial\boldsymbol{x}_k} = \boldsymbol{0}, \quad k = 1, 2, \ldots, N-1\,, \tag{2.54}$$

$$w_0\sum_i \frac{2C_{\lambda,i}}{(t_f - t_0)w_i}\boldsymbol{D}^*_{0i}\tilde{\boldsymbol{\lambda}}_i^{\intercal} + \frac{\partial\widetilde{\mathcal{L}}_0}{\partial\boldsymbol{x}_0} = -\boldsymbol{c}_0, \quad (k = 0)\,, \tag{2.55}$$

$$w_N\sum_i \frac{2C_{\lambda,i}}{(t_f - t_0)w_i}\boldsymbol{D}^*_{Ni}\tilde{\boldsymbol{\lambda}}_i^{\intercal} + \frac{\partial\widetilde{\mathcal{L}}_N}{\partial\boldsymbol{x}_N} = -\boldsymbol{c}_N, \quad (k = N)\,, \tag{2.56}$$

$$\frac{\partial\Phi}{\partial\boldsymbol{x}_0} + \boldsymbol{\nu}^{\intercal}\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}_0} + \frac{2C_{\lambda,0}}{(t_f - t_0)w_0}\tilde{\boldsymbol{\lambda}}_0^{\intercal} = \boldsymbol{c}_0, \quad (k = 0)\,, \tag{2.57}$$

$$\frac{\partial\Phi}{\partial\boldsymbol{x}_N} + \boldsymbol{\nu}^{\intercal}\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{x}_N} - \frac{2C_{\lambda,N}}{(t_f - t_0)w_N}\tilde{\boldsymbol{\lambda}}_N^{\intercal} = \boldsymbol{c}_N, \quad (k = N)\,, \tag{2.58}$$

$$\sum_k \frac{\partial\widetilde{\mathcal{H}}_k}{\partial t_0} + \frac{\partial\Phi}{\partial t_0} + \tilde{\boldsymbol{\nu}}^{\intercal}\frac{\partial\boldsymbol{\phi}}{\partial t_0} = 0\,, \tag{2.59}$$

$$\sum_k \frac{\partial\widetilde{\mathcal{H}}_k}{\partial t_f} + \frac{\partial\Phi}{\partial t_f} + \tilde{\boldsymbol{\nu}}^{\intercal}\frac{\partial\boldsymbol{\phi}}{\partial t_f} = 0\,, \tag{2.60}$$

where $\boldsymbol{c}_0$ and $\boldsymbol{c}_N$ are arbitrary real numbers, and $\boldsymbol{D}^*$ is a dual version of the differentiation matrix which applies to the covectors. This matrix is computed as [23, 29, 37]

$$\boldsymbol{D}^*_{ik} = -\frac{w_i}{w_k}\boldsymbol{D}_{ki}, \tag{2.61}$$

where w_i and w_k are Gaussian quadrature weights. And, because the optimal Hamiltonian is constant [9, 37], the summation terms in (2.59) and (2.60) can be simplified by

$$\sum_k \frac{\partial\widetilde{\mathcal{H}}_k}{\partial t_0} = -\sum_k \frac{\partial\widetilde{\mathcal{H}}_k}{\partial t_f} = -\mathcal{H}^{\dagger}\,. \tag{2.62}$$

Resulting in a satisfactory agreement of (2.59) and (2.60) with (2.34) and (2.35).

Because (2.49) is in discrete from, it is not possible to distinguish the terms on the inside of the Gaussian quadrature to the ones that are outside (as is the case with the integral terms in the continuous version of the problem). As a result, coupling is introduced between (2.55) and (2.56) with (2.57) and (2.58), respectively, as the endpoints of the trajectory are present on each pair of conditions. This coupling can be interpreted as a loss of information with respect to the original continuous optimal control problem [23, 29]. The *closure conditions* [23, 29]

$$\boldsymbol{c}_0 = 0\,, \tag{2.63}$$

$$\boldsymbol{c}_N = 0\,, \tag{2.64}$$

assert equivalence between the solution of the NLP with the solution of the boundary-value problem.

There are a few disadvantages of using direct methods. The major one is that there is some information loss due to the process of discretizing first and dualizing after. Ultimately, the problem being solved is not exactly the same as the original problem statement. Because of this, some post-processing tasks must be done in order to double check the validity of a solution. One of this tasks is to inspect the Hamiltonian and verify if it behaves as expected [37]. Despite this, the advantages of direct methods over indirect methods greatly outclass the disadvantages.

2.5 Covector Mapping

I order to assert an equivalence between the KKT conditions and the discrete Hamiltonian boundary-Value problem it is required to relate the KKT multipliers, $\tilde{\boldsymbol{\nu}}$, $\tilde{\boldsymbol{\lambda}}$ and $\tilde{\boldsymbol{\mu}}$, with the covectors, $\boldsymbol{\nu}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, respectively. This way, a solution to the KKT conditions is guaranteed to be a solution to the Hamiltonian boundary-value problem as well. Therefore, the two expressions (2.42) and (2.49), although syntactically distinct, must be equivalent. This equivalence can be expressed as

$$\mathcal{J}^{\lambda N} = \mathcal{J}^{N\lambda}\,, \tag{2.65}$$

where the order of the superscripts λ and N illustrates the order between the operations of dualization and discretization, respectively. Performing a little algebra leads to the following relationships:

$$\boldsymbol{\nu} = \tilde{\boldsymbol{\nu}}\,, \tag{2.66}$$

$$\mathcal{L}_k = \frac{2}{(t_f - t_0)w_k}\tilde{\mathcal{L}}_k\,. \tag{2.67}$$

And further expanding the terms in (2.67) yields:

$$\mathcal{H}_k = \frac{2}{(t_f - t_0)w_k}\tilde{\mathcal{H}}_k\,, \tag{2.68}$$

$$\boldsymbol{\lambda}_k = \frac{2C_{\lambda,k}}{(t_f - t_0)w_k}\tilde{\boldsymbol{\lambda}}_k\,, \tag{2.69}$$

$$\boldsymbol{\mu}_k = \frac{2C_{\mu,k}}{(t_f - t_0)w_k}\tilde{\boldsymbol{\mu}}_k\,. \tag{2.70}$$

Equations (2.66) through (2.70) are a generic form of the covector mapping theorem for linear mappings of the independent variable.

For visual aid, Fig. 2.3 presents a schematic of direct and indirect procedures, and also of the covector mapping theorem which bridges the gap between the two discrete versions of the problem. Going from left to right in the Figure is equivalent to deriving the optimality conditions, and going from top to bottom represents the operation of discretization. This way it is shown that in an indirect method the derivation of optimal conditions occurs first and the discretization occurs after, where as in a direct method the opposite is true. Because the operations of dualization and discretization are not commutative, the covector mapping is required to connect the discrete HBVP with the KKT conditions.

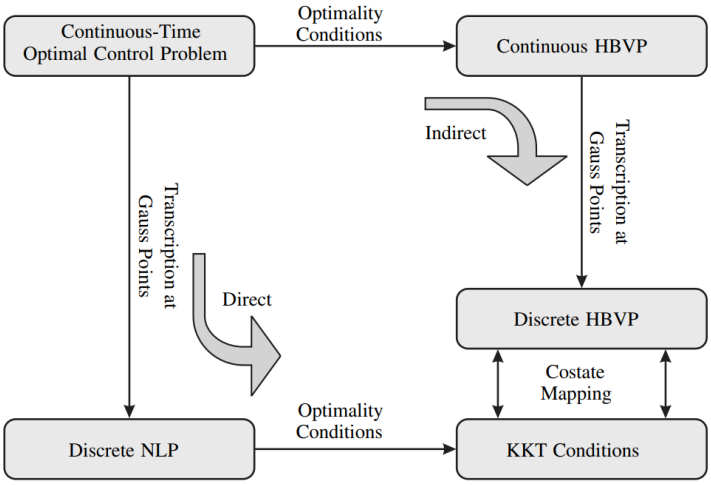


Figure 2.3: Commutative diagram between direct and indirect method routes to solve an optimal control problem [19].

Table 2.1 presents three alternative sets for the scalars $C_{\lambda,k}$ and $C_{\mu,k}$ that can be to used in the NLP problem, namely, the *minimal mapping*, the *normalizing mapping* and the *automatic mapping* sets. The minimal mapping set simplifies the computations in processing. By using this scalar set, the decision variables t_0 and t_f are moved from the denominator to the numerator in the case of the dynamic defect constraints, which simplifies the calculation of the derivatives with respect to these variables. The remaining constraints, i.e., the path constraints and the event constraints, maintain a simple form as no new decision variables are introduced into the problem. By omitting these variables whenever possible a few derivative computations can be spared, and considering that the derivatives with respect to t_f and t_0 are computed in every iterative loop in order to build the Jacobian matrix it is expected that the use of the minimal mapping results in faster iterations. However, the use of the minimal mapping set always requires some post-processing tasks, as the KKT multipliers need to be equated to the costates once the discrete solution to the NLP is found.

In contrast, the normalizing mapping set will introduce the variables t_f and t_0 into the path constraints which can potentially increase the iterations times, but the result is a relation between costates and KKT multipliers that is more uniform, as (2.69) and (2.70) become a simple quadrature weight normalization of the KKT multipliers. The normalizing mapping set does not require additional pre-processing jobs.

Finally, the automatic mapping set yields a direct relationship between the two covector types, and so post-processing jobs are reduced. The disadvantage of using this scalar set is that it introduces decision

Table 2.1: Sets of scalar factors to use in the NLP problem in order to obtain *minimal*, *normalizing* and *automatic* covector mappings.

	$C_{\lambda,k}$	$C_{\mu,k}$
Minimal mapping	$\frac{t_f - t_0}{2}$	1
Normalizing mapping	$\frac{t_f - t_0}{2}$	$\frac{t_f - t_0}{2}$
Automatic mapping	$\frac{t_f - t_0}{2} w_k$	$\frac{t_f - t_0}{2} w_k$

variables into the constraints that need not be there for an accurate solution of the primal variables, and so the Jacobian matrix becomes slightly more complex, which can potentially result in longer iteration times. There is also the introduction of the Gaussian quadrature weights, w_k, into the NLP which change according to the number of discretization points used which implies an increase in pre-processing jobs.

In summary, the covector mapping theorem bridges the gap between direct and indirect methods through a choice of scalar sets, $C_{\lambda,k}$ and $C_{\mu,k}$. The sets presented in Table 2.1 are the ones with the most obvious advantages, but other scalar sets can be used in the NLP, provided consistency is preserved.

Chapter 3

Implementation of the Flipped Radau Method for Multiphase Problems

3.1 Multiphase Optimal Control Problem

In this Section, the generic multiphase optimal control problem is presented. The formulation is based on the problem of Section 2.1, which can be interpreted as a single phase problem. The purpose of a generic multiphase problem formulation is to account for cases in which state discontinuities are expected. In the context of rocket launchers composed of multiple stages, state discontinuities are expected at stage separation events.

Letting $p \in [1, \ldots, P]$ be a scalar integer indicating a specific phase, where P is the total number of phases in the problem, a generic multiphase optimal control problem can be formulated as minimizing the cost functional

$$\mathcal{J} = \Phi\big(\boldsymbol{x}(t_0^{(1)}), t_0^{(1)}, \boldsymbol{x}(t_f^{(P)}), t_f^{(P)}\big) + \sum_{p=1}^{P} \left\{ \int_{t_0^{(p)}}^{t_f^{(p)}} \Psi\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big)\, \mathrm{d}t \right\}, \tag{3.1}$$

associated with the trajectory of the dynamic system

$$\dot{\boldsymbol{x}}^{(p)}(t) = \boldsymbol{f}^{(p)}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big), \qquad p = 1, \ldots, P\,, \tag{3.2}$$

subject to the path constraints

$$\boldsymbol{h}^{(p)}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \leq \boldsymbol{0}, \qquad p = 1, \ldots, P\,, \tag{3.3}$$

the event constraints

$$\boldsymbol{\phi}^{(p)}\big(\boldsymbol{x}(t_0^{(p)}), t_0^{(p)}, \boldsymbol{x}(t_f^{(p)}), t_f^{(p)}\big) \leq \boldsymbol{0}, \qquad p = 1, \ldots, P\,, \tag{3.4}$$

and the phase linkage conditions

$$\Delta\boldsymbol{x}_{\min}^{(p)} \;\leq\; \boldsymbol{\ell}^{(p)} = \boldsymbol{x}(t_0^{(p+1)}) - \boldsymbol{x}(t_f^{(p)}) \;\leq\; \Delta\boldsymbol{x}_{\max}^{(p)}, \qquad p = 1, \ldots, P-1\,, \tag{3.5}$$

where $\Delta\boldsymbol{x}_{\min}^{(p)}$ and $\Delta\boldsymbol{x}_{\max}^{(p)}$ are, respectively, the user-defined lower and upper boundary vectors for the linkage

conditions. The functions presented in (3.1) through (3.5) are defined by the following mappings:

$$\Phi : \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}\,, \tag{3.6}$$

$$\Psi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}\,, \tag{3.7}$$

$$\boldsymbol{f}^{(p)} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}\,, \tag{3.8}$$

$$\boldsymbol{h}^{(p)} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_h^{(p)}}\,, \tag{3.9}$$

$$\boldsymbol{\phi}^{(p)} : \mathbb{R}^{n_x} \times \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}^{n_\phi^{(p)}}\,, \tag{3.10}$$

$$\boldsymbol{\ell}^{(p)} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}\,, \tag{3.11}$$

where $n_{\boldsymbol{x}}$ is the dimension of the state vector, $n_{\boldsymbol{u}}$ is the dimension of the control vector, $n_{\boldsymbol{h}}^{(p)}$ is the number of simultaneous path constraints on phase p, and $n_{\boldsymbol{\phi}}^{(p)}$ is the total number of event constraints of phase p. This formulation indicates that the constraints are not explicit functions of time, therefore, the systems in study are time invariant. Also, the Mayer term and the Lagrange term are not expected to change from phase to phase, which is shown by the absence of the superscript "(p)" which would be attached to these functions in (3.1). Also, the state and control vectors must not change their sizes from phase to phase, again this is shown by the absence of the corresponding superscript "(p)" in $n_{\boldsymbol{x}}$ and $n_{\boldsymbol{u}}$.

Equation (3.5), which corresponds to the phase linkage conditions, consists of two distinct constraints, the upper bound and the lower bound of the eventual state discontinuities. This allows the specification of several discontinuity scenarios. Table 3.1 presents three possible scenarios of phase linkage conditions in order to describe a given state over a phase transition. The vector $\boldsymbol{a}$ represents a generic array that is arbitrary (user-defined), and the phase indicator, p, is omitted for syntax simplicity.

Table 3.1: Three different scenarios of phase linkage conditions in a given phase p.

	Relationship between $\Delta\boldsymbol{x}_{\text{min}}$ and $\Delta\boldsymbol{x}_{\text{max}}$	Note
1. Continuity	$\Delta\boldsymbol{x}_{\text{min}} = \Delta\boldsymbol{x}_{\text{max}} = 0$	
2. Known discontinuity	$\Delta\boldsymbol{x}_{\text{min}} = \Delta\boldsymbol{x}_{\text{max}} = \boldsymbol{a}$	$\boldsymbol{a} \in \mathbb{R}^{n_x}$
3. Unknown discontinuity	$\Delta\boldsymbol{x}_{\text{min}} < \Delta\boldsymbol{x}_{\text{max}}$	$\Delta\boldsymbol{x}_{\text{min}}, \Delta\boldsymbol{x}_{\text{max}} \in \mathbb{R}^{n_x}$

It is possible to enforce a continuous state, a known discontinuity or an unknown discontinuity. For instance, in a stage separation event where the ejected mass is a known parameter, $m_{\text{ejected}} = \text{known}$, the linkage condition for mass would be

$$\Delta m_{\text{min}} = \Delta m_{\text{max}} = -m_{\text{ejected}}\,, \tag{3.12}$$

thus enforcing a known discontinuity of mass at that point (case 2 in Table 3.1). It is up to the user to decide what the values $\Delta\boldsymbol{x}_{\text{min}}$ and $\Delta\boldsymbol{x}_{\text{max}}$ should be in order to best represent a given problem.

3.2 Multiphase NLP and Covector Mapping

Following the reasoning exposed in Section 2.4.1 for the single phase case, the NLP for multiphase problems can be easily worked out. The multiphase NLP is quite similar to the single phase counterpart in

that the multiphase NLP is just an aggregate of several single phase problems. The main difference is that the multiphase NLP contains additional *phase linkage conditions* required to relate one phase to the next. In a multiphase problem state discontinuities are expected, and so the phase linkage conditions are important in order to relate the states at the endpoints of each phase. The multiphase NLP can be expressed as minimizing the cost

$$\mathcal{J}^N = \Phi + \sum_{p=1}^{P} \frac{t_f^{(p)} - t_0^{(p)}}{2} \sum_{k}^{N^{(p)}} w_k^{(p)} \Psi_k \,, \tag{3.13}$$

subject to:

$$C_{\lambda,k}^{(p)} \left[\boldsymbol{f}_k^{(p)} - \frac{2}{t_f^{(p)} - t_0^{(p)}} \sum_i \boldsymbol{D}_{ki}^{(p)} \boldsymbol{x}_i^{(p)} \right] = \boldsymbol{0} \,, \tag{3.14}$$

$$C_{\mu,k}^{(p)} \boldsymbol{h}_k^{(p)} \leq \boldsymbol{0} \,, \tag{3.15}$$

$$\boldsymbol{\phi}^{(p)} \leq \boldsymbol{0} \,, \tag{3.16}$$

$$\Delta \boldsymbol{x}_{\text{min}}^{(p)} \leq \boldsymbol{\ell}^{(p)} \leq \Delta \boldsymbol{x}_{\text{max}}^{(p)} \,, \tag{3.17}$$

where $N^{(p)}$ is the number of *collocation points* used to discretize phase p.

Similarly, the covector mapping can also be generalized for multiphase based on the single phase formulation presented in Section 2.5. Multiphase covector mapping can be written as:

$$\boldsymbol{\lambda}_k^{(p)} = \frac{2C_{\lambda,k}^{(p)}}{(t_f^{(p)} - t_0^{(p)}) w_k} \tilde{\boldsymbol{\lambda}}_k^{(p)} \,, \tag{3.18}$$

$$\boldsymbol{\mu}_k^{(p)} = \frac{2C_{\mu,k}^{(p)}}{(t_f^{(p)} - t_0^{(p)}) w_k} \tilde{\boldsymbol{\mu}}_k^{(p)} \,, \tag{3.19}$$

$$\boldsymbol{\nu}^{(p)} = \tilde{\boldsymbol{\nu}}^{(p)} \,. \tag{3.20}$$

Again, the superscript $p = 1, 2, \ldots, P$ serves as an ordinal indicator of a given phase, in a problem with P sequential phases. Figure 3.1 illustrates the pattern of collocation nodes for a discretization of the domain composed of multiple phases using the flipped Radau pseudospectral method. The multiphase discretization scheme is a simple concatenation of several "single phase" schemes. It can be noticed that the "density" of nodes increases near the phase transition points.

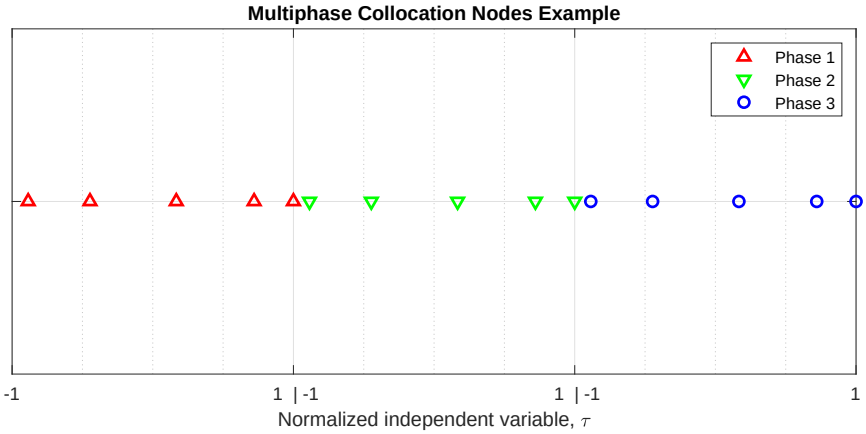


Figure 3.1: Illustration of the pattern of collocation nodes for multiple phase transcription. Example of three phases with five collocation nodes in each phase.

3.2.1 Scalar set selection for multiphase covector mapping and simplified NLP

In this work the *minimal mapping set* is used because it does not require dedicated pre-processing tasks. Further, this mapping seems to be the standard in literature [9, 23]. The multiphase version of the minimal mapping set from Table 2.1 is shown in Table 3.2.

Table 3.2: Multiphase scalar factors for the *minimal* costate mapping.

	$C_{\lambda,k}^{(p)}$	$C_{\mu,k}^{(p)}$
Choice of scalar factors for costate mapping	$\frac{t_f^{(p)} - t_0^{(p)}}{2}$	1

The format of the minimal mapping scalar factors for multiphase is similar to the case of single phase seen in Table 2.1, with the added superscript "(p)" to the variables t_0 and t_f in order to specify the phase, as each phase contains its own pair of initial and final times. With the mapping selection shown in Table 3.2 the constraints from (3.14) and (3.15) can be simplified to:

$$\boldsymbol{\xi}_k^{(p)} = \frac{t_f^{(p)} - t_0^{(p)}}{2} \boldsymbol{f}_k^{(p)} - \sum_i \boldsymbol{D}_{ki}^{(p)} \boldsymbol{x}_i^{(p)} = \boldsymbol{0}\,, \tag{3.21}$$

$$\boldsymbol{h}_k^{(p)} \leq \boldsymbol{0}\,, \tag{3.22}$$

where the vector $\boldsymbol{\xi}_k^{(p)}$ is a shortened representation of the constraints for dynamic defects from (3.21) for phase p. Equations (3.18) and (3.19) will also be subject to simplifications once the scalars $C_{\lambda,k}^{(p)}$ and $C_{\mu,k}^{(p)}$ are substituted with the respective values from Table 3.2, these equations become, respectively:

$$\boldsymbol{\lambda}_k^{(p)} = \frac{\tilde{\boldsymbol{\lambda}}_k^{(p)}}{w_{k^{(p)}}}\,, \tag{3.23}$$

$$\boldsymbol{\mu}_k^{(p)} = \frac{2}{\left(t_f^{(p)} - t_0^{(p)}\right)} \frac{\tilde{\boldsymbol{\mu}}_k^{(p)}}{w_{k^{(p)}}}\,, \tag{3.24}$$

Ultimately, the discrete form of the cost functional expressed in (3.13) and the discrete form of the constraints expressed in (3.16), (3.17), (3.21) and (3.22) are the ones to send to the nonlinear solver.

3.3 Vector Formatting and Jacobian Matrix

3.3.1 Input format of the nonlinear solver

In order to solve an optimal control problem via the Radau pseudospectral method, one needs to transcribe the original problem statement into a format which a nonlinear solver can interpret. A nonlinear program is expressed as [26–28] minimizing

$$J(\boldsymbol{X}_{\text{NLP}})\,, \tag{3.25}$$

subject to

$$\boldsymbol{C}_{\min} \leq \boldsymbol{C}(\boldsymbol{X}_{\text{NLP}}) \leq \boldsymbol{C}_{\max}\,, \tag{3.26}$$

$$\boldsymbol{X}_{\min} \leq \quad \boldsymbol{X}_{\text{NLP}} \quad \leq \boldsymbol{X}_{\max}\,. \tag{3.27}$$

where $\boldsymbol{C}_{\max}$ and $\boldsymbol{C}_{\min}$ are, respectively, the upper and lower boundary vectors of the constraints, and $\boldsymbol{X}_{\max}$ and $\boldsymbol{X}_{\min}$ are the upper and lower boundary vectors of the decision variables, respectively. Essentially, there is a cost function, $J(\boldsymbol{X}_{\text{NLP}})$, to be minimized, a list of algebraic constraints, $\boldsymbol{C}(\boldsymbol{X}_{\text{NLP}})$, to be satisfied and a list of parameters, $\boldsymbol{X}_{\text{NLP}}$, that act as decision variables. In order to transform the optimal control problem into a NLP it is necessary to concatenate all the decision variables into a main vector and to aggregate all constraints into a main constraint vector, this is depicted by vectors $\boldsymbol{X}_{\text{NLP}}$ and $\boldsymbol{C}(\boldsymbol{X}_{\text{NLP}})$ respectively. The algebraic constraints contained in $\boldsymbol{C}(\boldsymbol{X}_{\text{NLP}})$ can assert either an equality or an inequality by setting $\boldsymbol{C}_{\max} = \boldsymbol{C}_{\min}$ or $\boldsymbol{C}_{\max} > \boldsymbol{C}_{\min}$ respectively.

3.3.2 Formatting the vector of decision variables

In order to construct a nonlinear programming problem one needs both, a list of decision variables and a list of algebraic constraints. Let

$$\boldsymbol{XU}^{(p)} = \begin{bmatrix} \boldsymbol{x}_0^{(p)\mathsf{T}} & \boldsymbol{x}_1^{(p)\mathsf{T}} & \boldsymbol{u}_1^{(p)\mathsf{T}} & \dots & \boldsymbol{x}_i^{(p)\mathsf{T}} & \boldsymbol{u}_i^{(p)\mathsf{T}} & \dots & \boldsymbol{x}_{N^{(p)}}^{(p)\mathsf{T}} & \boldsymbol{u}_{N^{(p)}}^{(p)\mathsf{T}} \end{bmatrix}^{\mathsf{T}}, \tag{3.28}$$

be the concatenated decision vector of state and control associated with the nodes of a given phase p. Notice that there is no control variable associated with the very first node $i = 0$, this is because this node is not collocated [51]. The resulting size of this column vector is $[n_{\boldsymbol{x}} + N^{(p)}(n_{\boldsymbol{x}} + n_{\boldsymbol{u}})] \times 1$.

The complete vector of decision variables can be expressed as:

$$\boldsymbol{X}_{\text{NLP}} = \begin{bmatrix} \boldsymbol{XU}^{(1)\mathsf{T}} & \dots & \boldsymbol{XU}^{(p)\mathsf{T}} & \dots & \boldsymbol{XU}^{(P)\mathsf{T}} & t_f^{(1)} & \dots & t_f^{(p)} & \dots & t_f^{(P)} \end{bmatrix}^{\mathsf{T}} \tag{3.29}$$

resulting in a column vector of size $\left[\sum_{p=1}^{P}[n_{\boldsymbol{x}} + N^{(p)}(n_{\boldsymbol{x}} + n_{\boldsymbol{u}})] + P\right] \times 1$. The lower and upper boundaries of the vector of decision variables, $\boldsymbol{X}_{\min}$ and $\boldsymbol{X}_{\max}$ can be build according to an analogous procedure, given the parameters specified by the user for each problem.

3.3.3 Constraints formatting

The cost functional fed to the nonlinear solver is identical to the cost described in (3.13), thus,

$$J(\boldsymbol{X}_{\text{NLP}}) = \mathcal{J}^N. \tag{3.30}$$

In order to construct the concatenated vector, $\boldsymbol{C}(\boldsymbol{X}_{\text{NLP}})$, two additional arrays are introduced, namely $\boldsymbol{F}^{(p)}$ and $\boldsymbol{H}^{(p)}$, containing all the constraints corresponding to phase p from (3.21) and (3.22):

$$\boldsymbol{F}^{(p)} = \begin{bmatrix} \boldsymbol{\xi}_1^{(p)\mathsf{T}} & \boldsymbol{\xi}_2^{(p)\mathsf{T}} & \dots & \boldsymbol{\xi}_k^{(p)\mathsf{T}} & \dots & \boldsymbol{\xi}_{N^{(p)}}^{(p)\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \tag{3.31}$$

$$\boldsymbol{H}^{(p)} = \begin{bmatrix} \boldsymbol{h}_1^{(p)\mathsf{T}} & \boldsymbol{h}_2^{(p)\mathsf{T}} & \dots & \boldsymbol{h}_k^{(p)\mathsf{T}} & \dots & \boldsymbol{h}_{N^{(p)}}^{(p)\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \tag{3.32}$$

where $\boldsymbol{\xi}_k^{(p)}$ and $\boldsymbol{h}_k^{(p)}$ are as in (3.21) and (3.22), respectively. This way, the concatenated vector of constraints can be written as

$$\boldsymbol{C}(\boldsymbol{X}_{\text{NLP}}) = \begin{bmatrix} \boldsymbol{F}^{(1)\mathsf{T}} & \dots & \boldsymbol{F}^{(P)\mathsf{T}} & \boldsymbol{H}^{(1)\mathsf{T}} & \dots & \boldsymbol{H}^{(P)\mathsf{T}} & \boldsymbol{\phi}^{(1)\mathsf{T}} & \dots & \boldsymbol{\phi}^{(P)\mathsf{T}} & \boldsymbol{\ell}^{(1)\mathsf{T}} & \dots & \boldsymbol{\ell}^{(P-1)\mathsf{T}} \end{bmatrix}^{\mathsf{T}}. \tag{3.33}$$

The lower and upper boundaries for the vector of constraints, $\boldsymbol{C}_{\text{min}}$ and $\boldsymbol{C}_{\text{max}}$, are constructed in an analogous manner, with the peculiarity that the boundaries corresponding to the dynamic defects, must be set invariably equal to zero, $\boldsymbol{F}^{(1:P)} = \boldsymbol{0}$, such that the dynamics are enforced and that the trajectory satisfies the equations of motion at all times.

3.3.4 Jacobian Matrix

The Jacobian matrix is a quantitative description of the derivatives of the constraints with respect to the decision variables, providing a first order gradient that is necessary to aid the nonlinear solver. This section serves to overview the components of the Jacobian matrix that results from a discretization based on the flipped Radau method. An in-depth description of this matrix can be consulted in Appendix A.

Each row of the Jacobian matrix corresponds to an algebraic constraint, and each column corresponds to a decision variable. Ultimately, each element represents a linear dependency between a constraint (row) with respect to a decision variable (column). Mathematically the Jacobian can be expressed as [51]

$$\textbf{Jac} = \nabla_{\boldsymbol{X}_{\text{NLP}}} \begin{bmatrix} J(\boldsymbol{X}_{\text{NLP}}) \\ \boldsymbol{C}(\boldsymbol{X}_{\text{NLP}}) \end{bmatrix} = \begin{bmatrix} \nabla \mathcal{J}^{N} \\ \nabla \boldsymbol{F}^{(1:P)} \\ \nabla \boldsymbol{H}^{(1:P)} \\ \nabla \boldsymbol{\phi}^{(1:P)} \\ \nabla \boldsymbol{\ell}^{(1:P-1)} \end{bmatrix} . \tag{3.34}$$

The format of the vectors $\boldsymbol{X}_{\text{NLP}}$ and $\boldsymbol{C}(\boldsymbol{X}_{\text{NLP}})$ dictate the sparsity pattern of the Jacobian matrix. Figure 3.2 is a visual representation of the pattern of the Jacobian for an example problem with four phases. In this Figure, zero elements are represented by white space, while non-zero elements are represented with coloured dots. The first row of the matrix corresponds to the discrete cost functional where the magenta circles represent a Lagrangian cost and the blue dot at the end of phase four represents a Mayer cost; there are four open terminal times (green columns); a scalar path constraint applied to all phases (concatenated diagonals in cyan); vector event constraints at the terminal times of every phase (magenta blocks); and three linkage conditions connecting the four phases sequentially (red diagonals at the bottom rows). The four phases can be distinguished by the four large red blocks with blue diagonal smaller blocks (corresponding to the dynamic defects of each phase). The prominent red diagonals in the Jacobian correspond to the sparsity pattern of the differentiation matrices and to the identity matrices corresponding to the linkage conditions. The values of the Jacobian matrix corresponding to the red dots are static, i.e., they will remain unchanged in every iteration loop.

It is relevant to note that the diagonals on the linkage conditions rows are, invariably, positive and negative identity matrices with the dimension of the state vector, $\pm \boldsymbol{I}_{n_x \times n_x}$, connecting one phase to the next. Also, it is noticeable in Fig. 3.2, that the constraints of different phases are decoupled from one another, in other words, decision variables of one phase do not affect the constraints of any other phase, thus the resulting Jacobian matrix is very sparse.

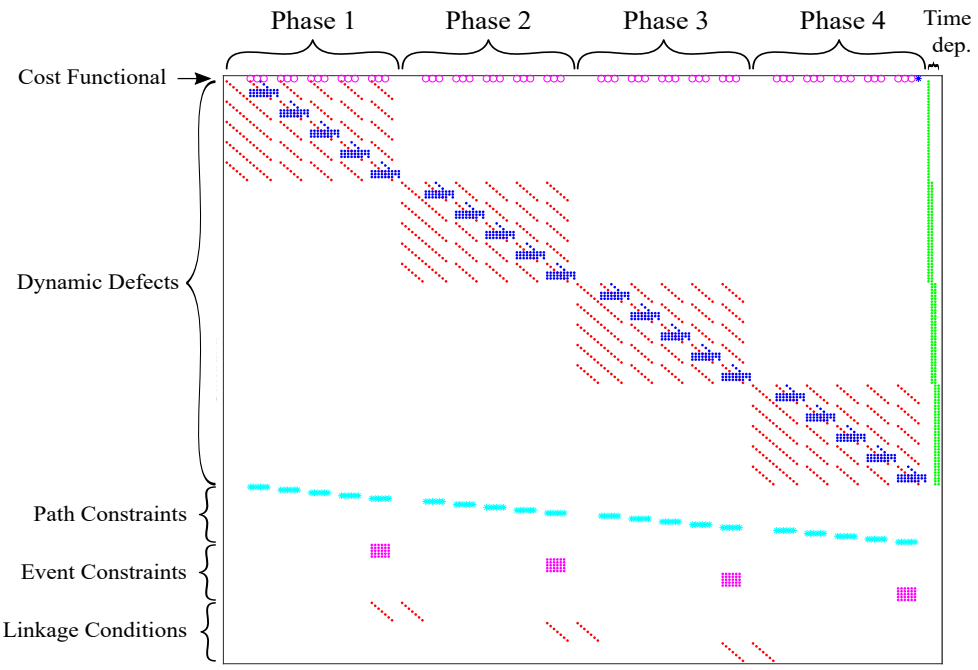


Figure 3.2: Sparsity pattern of the Jacobian matrix for an example problem with four phases, five collocation points on each phase, four open terminal times, one scalar path constraint applying to every phase and a terminal constraint vector applying to every phase. White spacing represents zero-valued elements.

3.4 Overview of the Solving Procedure

In order to solve a given optimal control problem a procedural approach is taken. This procedure needs to be generic in order to be able to handle as many problems as possible. The main task at hand is to perform a transformation of the user input into variables and constraints that are useful to feed an NLP solver. The following description applies to the MATLAB tool SPARTAN developed at DLR [32, 51]. Figure 3.3 presents a high-level overview of the solving procedure implemented in SPARTAN for visual aid.

1. User Input First the user provides a set of MATLAB functions, with everything deemed necessary to describe the problem, namely: the Mayer term and the Lagrange term, the right-hand side of the equations of motion, the algebraic function of path constraints (and the respective upper and lower boundaries), the algebraic function of the event constraints (and the respective upper and lower boundaries), the upper and lower boundaries of the endpoint variables of every phase, the upper and lower boundaries of state and control along the trajectory of every phase, and the upper and lower boundaries of the phase linkage conditions of every phase transition. Also, with regards to discretization rules, the user needs to provide the number of phases in the problem, the number of collocation nodes to use on each phase and the concrete number of states and controls of the problem. Finally the user may choose the NLP solver and the differentiation method along with some minor solver options.

2. Transcription Process Once the user provides the input MATLAB functions (along with the respective handle names), then SPARTAN takes control and starts the transcription process. This process consists on generating the location of the collocation points (roots of the flipped Radau polynomial) according to the number of nodes specified by the user; generating the concatenated vectors of upper and

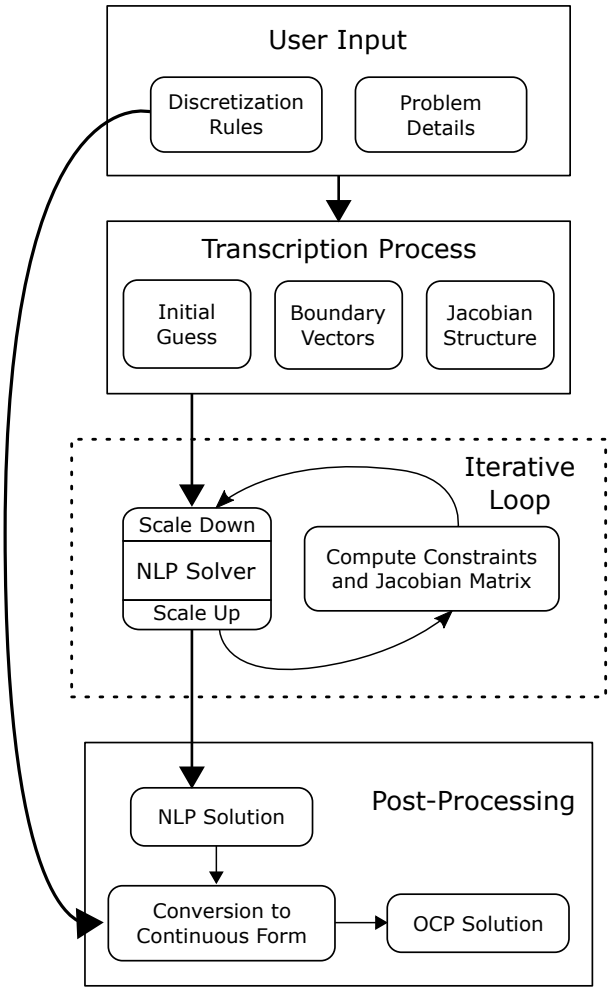


Figure 3.3: Flowchart of the solving process by the MATLAB tool SPARTAN.

lower boundaries of the decision variables and of the algebraic constraints based on the problem data provided by the user, perform a dependency analysis on the algebraic function provided by the user in order to construct the structure of the Jacobian matrix with satisfactory sparsity; and generate an initial guess of state and control at the collocation nodes in the form of a concatenated vector of decision variables.

3. Iterative Loop The initial guess vector and the boundary vectors then scaled and fed to the NLP solver where an iterative loop takes place. For all purposes the solver can be treated as a "black box" which, at every iteration, provides an update on the decision variables and requests an update on both, the constraints and the respective Jacobian matrix. Because the user provides the algebraic functions in "full size", the updated variables need to be scaled up in every iteration loop in order to compute the constraints, and the constraints and the Jacobian need to be scaled down in every iteration loop before being sent to the solver.

4. Post-processing Finally, if the problem is feasible, the solver yields a solution to the NLP, which consists on the concatenated vector of decision variables. Some post-processing tasks are then required in order to output a solution that makes sense to the user. The discretization rules provided by the user are taken in order to "break apart" the vector of decision variables into states, controls and endpoint times.

A similar process takes place in order to extract the dual variables. Then, one needs to extrapolate the endpoint controls and costates corresponding to the initial time, t_0, of each phase, as this endpoint is not collocated. After that, the costate mapping is performed and the Hamiltonian is computed. Finally a "continuous" solution is generated based on the Lagrange interpolating polynomials. This interpolation is applied to state, control, costate and Hamiltonian alike.

Chapter 4

Test and Validation with Numerical Examples

4.1 Problem 1: Falcon 9 Rocket Boost-Back Burn and Vertical Landing (3 phases, 2-D trajectory)

The first problem to consider is an adaptation from [67]. This example is concerned with the recovery of the first stage of an orbital launcher via vertical landing, where the landing target is located close to the launching site (return-to-launch-site scenario). The vehicle in question is based on the characteristics of the main booster of SpaceX's Falcon 9 rocket which is illustrated in Figure 4.1. This problem is appropriate to validate the algorithm developed in this work because it is composed of three distinct phases, resulting from the mutation of the equations of motion from one phase to the next. The algorithm is validated by analysis of the optimality conditions.

The problem occurs in a vertical, two-dimensional plane (2-D). At the beginning of the trajectory it is assumed that stage separation has already occurred and that the booster has performed the "turn back"

Figure 4.1: Illustration of the main stage of SpaceX's Falcon 9 rocket [68].

attitude manoeuvre such that the boost-back-burn may begin. The trajectory is composed of three phases:

Phase 1 Boost-back. At $t = t_0$ the rocket is in mid air with ascending linear momentum. Three engines (out of nine) ignite and the thrust direction is constrained to point strictly horizontally. In this phase the rocket inverts its horizontal motion. The duration of the boost-back burn, $t = t_1$, is a known, fixed parameter.

Phase 2 Coast arc. At $t = t_1$ the engines shut down and remain off during all of phase 2. The vehicle takes a parabolic flight profile during this phase. It is taken for granted that an attitude justification manoeuvre takes place during this phase in order to roughly align the thrust vector with the velocity vector at the beginning of phase 3.

Phase 3 Landing. At $t = t_2$ one engine (out of nine) ignites. The vehicle is now controllable and the booster makes its way to the target site. Touch-down happens at $t = t_f$. Both t_2 and t_f are unknown variables to be determined.

The equations of motion are formulated in the target-centred reference frame [67]. Throughout the trajectory, both position and velocity are modelled in Cartesian coordinates, with the reference frame of displacement centred at the landing target location. Figure 4.2 presents a free-body diagram of the vehicle for visual reference. Downrange is represented by the horizontal axis, x, increasing from left to right, and altitude is represented by the vertical axis, y, increasing from bottom to top. The angle θ indicates the direction of the thrust vector, $\boldsymbol{T}$, and the flight-path angle, represented by the angle γ, indicates the direction of the velocity vector, $\boldsymbol{v}$. Both angles are measured from the x axis and positive in the direction of increasing y. The Cartesian components of velocity, v_x and v_y, are positive in the direction of increasing respective position coordinates. The aerodynamic drag force is represented by vector $\boldsymbol{D}$ which is always collinear with the velocity vector, and m is the mass of the vehicle. The Earth is assumed to be flat and non rotating, and the acceleration of gravity is assumed to be invariant with altitude throughout the trajectory, taking the value g_0. The control variable is the thrust tilt angle, θ. A point-mass approximation of the vehicle is employed, thus the attitude of the spacecraft is not modelled and the angle of attack is assumed to be zero at all times. An exponential model is used for the drag force. The optimal control problem can be formulated as minimizing the cost

$$\mathcal{J} = \Phi\big(m(t_f)\big) = -m(t_f)\,, \tag{4.1}$$

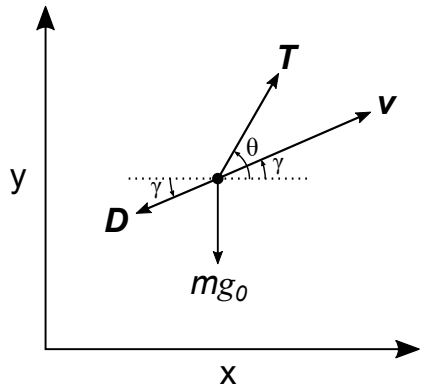


Figure 4.2: Free-body diagram of the vehicle.

associated with the dynamic system

$$\dot{x} = v_x \,, \tag{4.2}$$

$$\dot{y} = v_y \,, \tag{4.3}$$

$$\dot{v_x} = \frac{T}{m} k \cos\theta - \frac{D}{m} \cos\gamma \,, \tag{4.4}$$

$$\dot{v_y} = \frac{T}{m} k \sin\theta - \frac{D}{m} \sin\gamma - g_0 \,, \tag{4.5}$$

$$\dot{m} = -\frac{T}{I_{sp} g_0} k \tag{4.6}$$

and subject to the event constraints

$$x(t_0) = 36.022\,\text{km} \,, \tag{4.7}$$

$$y(t_0) = 60.708\,\text{km} \,, \tag{4.8}$$

$$v_x(t_0) = 1.052\,\text{km}\,\text{s}^{-1} \,, \tag{4.9}$$

$$v_y(t_0) = 1.060\,\text{km}\,\text{s}^{-1} \,, \tag{4.10}$$

$$m(t_0) = 76\,501\,\text{kg} \,, \tag{4.11}$$

$$x(t_f) = 0\,\text{km} \,, \tag{4.12}$$

$$y(t_f) = 0\,\text{km} \,, \tag{4.13}$$

$$v_x(t_f) = 0\,\text{km}\,\text{s}^{-1} \,, \tag{4.14}$$

$$-0.5\,\text{m}\,\text{s}^{-1} \leq v_y(t_f) \leq 0.5\,\text{m}\,\text{s}^{-1} \,, \tag{4.15}$$

$$m(t_f) = \text{free} \,, \tag{4.16}$$

with

$$D = \frac{1}{2} \rho_0 \exp\left\{ -\frac{y}{h_0} \right\} C_D \pi \frac{d^2}{4} v^2 \,, \tag{4.17}$$

$$v^2 = v_x^2 + v_y^2 \,, \tag{4.18}$$

$$\cos\gamma = \frac{v_x}{v} \,, \tag{4.19}$$

$$\sin\gamma = \frac{v_y}{v} \,, \tag{4.20}$$

and

$$T = \begin{cases} \frac{1}{3} T_{LO} & t \in [t_0, t_1] \\ 0 & t \in [t_1, t_2] \\ \frac{1}{9} T_{LO} & t \in [t_2, t_3] \end{cases} \qquad k = \begin{cases} 1 & t \in [t_0, t_1] \\ 0 & t \in [t_1, t_2] \\ 1 & t \in [t_2, t_3] \end{cases} \,, \tag{4.21}$$

where k is the engine throttle, ρ_0 is the air density at sea level, h_0 is the density scale height, C_D is the drag coefficient, d is the diameter of the vehicle and v is the norm of the velocity vector. The linkage conditions are omitted because there are no expected jump discontinuities on the sates of the systems in any phase transition. Table 4.1 shows the constants and parameters used in this problem.

The problem at hand is not formulated exactly the same way as in the original article, therefore, a few remarks are in order to relate the two formulations of this problem. Briefly stated, the problem implemented in this work contains fewer degrees of freedom than that of the original article. Namely:

1. The duration of the boost-back burn, t_1, is fixed, while in [67] this parameter is taken as an additional decision variable. The fixed value of engine shut down time of phase 1, t_1, was selected in such a way as to facilitate a vertical flight-path angle towards the end of phase 3.
2. The throttle level is fixed at the maximum value during phases 1 and 3, while in [67] the throttle is allowed vary between 0 and 1 in both of these phases.

Table 4.1: Relevant constants and parameters for the Falcon 9 first stage recovery problem.

Constant	Value	Unit
Specific impulse, I_{sp}	282	s
Lift-off Thrust, T_{LO}	5886	kN
Rocket diameter, d	3.66	m
Drag Coefficient, C_D	0.75	1
Density scale height, h_0	7500	m
Gravity acceleration at seal level, g_0	9.80665	$\mathrm{m\,s^{-2}}$
Air density at sea level, ρ_0	1.225	$\mathrm{kg\,m^{-3}}$
Phase 1 initial time, t_0	0	s
Phase 1 to phase 2 transition time, t_1	40.8	s
Dry mass, m_{dry}	25600	kg

3. Downrange, altitude, horizontal velocity at the final time of the trajectory are constrained to zero, while in [67] these variables appear solely as weighted minimization parameters in the cost functional.

Also, some parameters have been relaxed in the present formulation with respect to the original article. Namely:

1. The control is expressed in polar coordinates throughout the whole trajectory, therefore there is no need to include an additional path constraint in order to assert the modulus of Cartesian components. This contrasts with the formulation in [67] where the use of polar and Cartesian coordinates is alternated from phase to phase.
2. In the present formulation, and opposed to the original article, the landing phase is not constrained to a gravity turn, allowing extra manoeuvrability of the vehicle during this phase end facilitating a vertical flight path angle at the landing point.
3. The vertical component of the velocity at the landing time, $v_y(t_f)$, is allowed to vary as shown in (4.15), while in [67] this variable is a taken as a Mayer cost to be minimized.
4. The cost functional is composed only of the final mass of the vehicle, while in [67] the final positions and velocities are weighted in as well.

During phase 1 the trajectory is completely determined because the control sequence is known and the endpoint times are fixed, and in phase 2 the vehicle is uncontrollable due to the throttle being constrained to zero. Ultimately, the only degrees of freedom available to optimize the trajectory, therefore, are the engine reignition time, t_2, the total time of flight, t_f and the direction of the thrust vector during phase 3.

With this simplified formulation one makes sure that the problem is feasible, therefore an optimal solution is guaranteed to exist and it is possible to focus on the ability of the algorithm to find it. A simplified formulation of the problem is adequate in this case because there is no interest in studying the feasibility of the problem, there is only interest in validating the flipped Radau method.

This problem was solved using 10, 8 and 12 collocation nodes in phases 1, 2 and 3, respectively. The NLP solver used was IPOPT [28], and the method used to compute partial derivatives was the complex step differentiation method [48, 49]. The results are depicted in Figs. 4.3 through 4.12. Specifically, Figs. 4.3

through 4.9 deal with the solution of state and control, while Figs. 4.10 through 4.12 deal with the solution of the dual variables (covectors and Hamiltonian). In addition the resulting parameters of unknown times and final mass are presented in Table 4.2 including the references of the original article.

Figures 4.3 and 4.4 illustrate the Cartesian components of position, and the Cartesian components of velocity with time, respectively. The phase transition points are visible through the increase in density of collocation nodes. With regards to Fig. 4.4, the phase transitions are clearly pronounced due to the removable discontinuities present in each component. The horizontal component of velocity decreases linearly during the first phase and goes from positive to negative, inverting the direction of flight, which is indicative of the boost-back burn phase. This component stays constant during the coasting phase, as expected. It is noticeable that during phases 1 and 2, the vertical component of velocity decreases linearly, without noticeable discontinuities, indicating that this component has been subject to a constant acceleration during these two phases, undoubtedly the acceleration of gravity. It is also visible that the vertical component of velocity goes from positive to negative close to the $110\,\mathrm{s}$ mark, inverting the direction flight. Both velocity components go to zero during the landing phase, as expected. In contrast, regarding Fig. 4.3, the position does not contain removable discontinuities, this is due to the fact that position is the integration of velocity through time, resulting in a curve that is "one degree" smother. One relevant remark is that both the downrange and the altitude are tangent to the time axis at the endpoint of the trajectory, indicating a smooth landing.

The mass profile of the vehicle can be seen in Fig. 4.5. Due to the constant throttle level, and the constant specific impulses, the mass of the vehicle decays linearly during phases 1 and 3. The mass flow-rate is higher in phase 1 due to the larger thrust associated — there are three engines on during phase 1 and only one engine ignited during phase 3. Not remarkably, the mass stays constant during phase 2 (coast phase).

The angle indicating the thrust direction (control) is presented in Fig. 4.6. It can be seen that the direction of thrust is constrained to $180°$ during phase 1, and constrained to $0°$ during phase 2 (during phase 2 the thrust direction is inconsequential due to the throttle being constrained to zero). Finally, in the landing phase

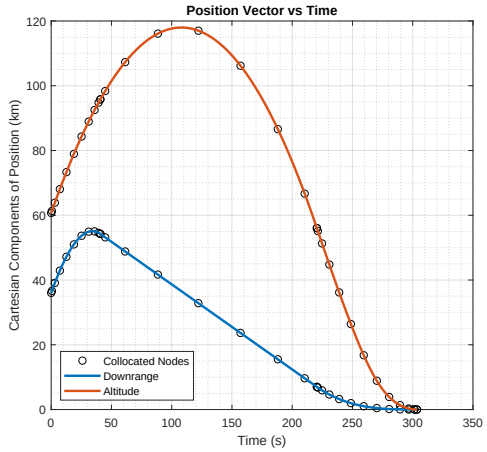


Figure 4.3: Downrange and altitude with time. Both position coordinates are tangent to the time axis at the final time.

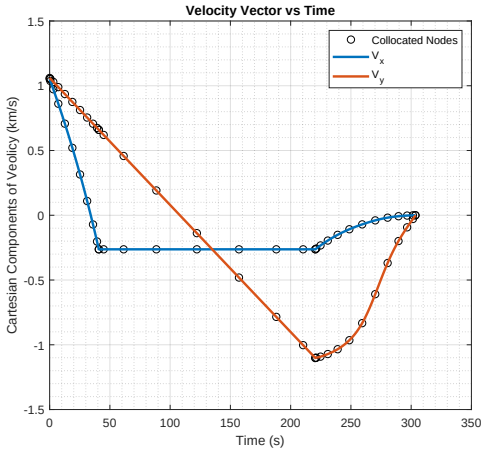


Figure 4.4: Cartesian components of velocity with time. Both components approach zero at the final time.

the direction of thrust is allowed to vary, and the angle draws a curve that approaches 90°, indicating that the thrust points vertically at the end of the trajectory.

Figure 4.7 shows the evolution of flight path angle of the vehicle in time. The flight path angle, representing the direction of the velocity vector, starts close to 45° at the beginning of the boost back phase, indicating an ascending trajectory compatible with the initial conditions expressed in (4.9) and (4.10). The Figure also shows the removable discontinuities occurring at the phase transition times, which goes in agreement with the discussion above with regards to the velocity components in Fig. 4.4, in particular, the slight discontinuity in the flight path angle occurring in the transition between coast phase and landing phase indicates that the powered landing does not follow a gravity turn (in a gravity turn the flight path angle is unaffected by the thrust), however, this angle does approach 270° in the last phase, indicating a vertical landing. Ultimately, Fig. 4.7 concurs with Fig. 4.6 during the landing phase in the sense that the velocity vector and the thrust vector both tend to be vertical at the end of the trajectory and point in opposite directions.

The trajectory of the vehicle is shown in Fig. 4.8, where the downrange and the altitude are plotted against each other. The boost-back phase is shown in blue, the coasting arc is shown in orange and the landing phase

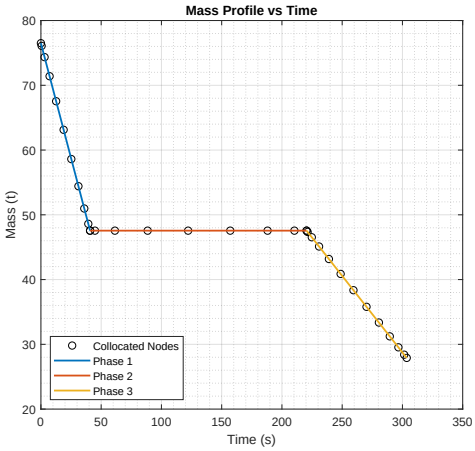


Figure 4.5: Profile of the total mass of the vehicle with time. Linear decay in phases 1 and 3 indicates constant mass flow-rate.

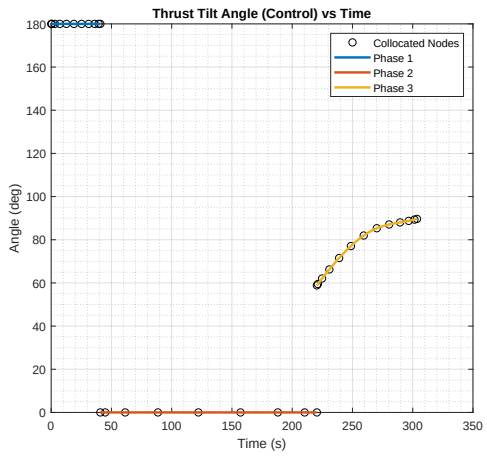


Figure 4.6: Thrust direction, θ, with time. Thrust direction approaches 90° at the final time.

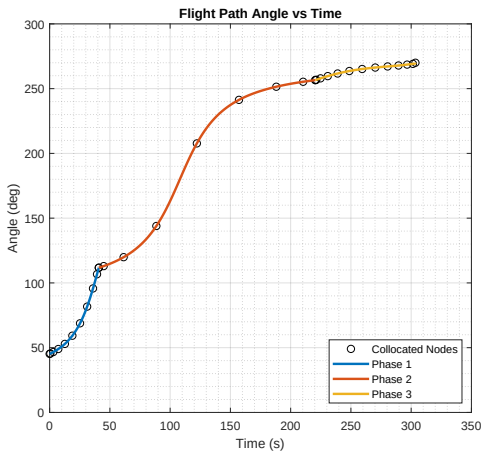


Figure 4.7: Flight path angle, γ, with time; flight path angle approaches 270° at the final time.

is represented in yellow. Curiously, the vehicle traces a path that vaguely resembles a shepherd's staff. The plot clearly shows that at the beginning of the trajectory the vehicle is travelling from left to right and with ascending velocity. The boost-back burn inverts the horizontal motion of the vehicle and during the coasting arc, the vehicle takes a parabolic flight due to being in complete free-fall. Finally, in the landing phase the vehicle approaches the zenith of the target and the flight path angle gets closer and closer to being vertical. Intuitively speaking, the trajectory follows a predictable path.

Fig. 4.9 presents a comparison between the trajectory yielded by SPARTAN and the reference solution [67]. It can be seen that the trajectories are quite similar, having an identical maximum altitude. The solutions diverge slightly during the descent of the vehicle, and the trajectory from SPARTAN approaches the target at a steeper flight path angle. This Figure is presented in order to confirm the plausibility of the scenario, given that the two problem formulations are similar. However, the two trajectories must be judged independently for their optimality, as the corresponding problem formulations are in fact not the same, having a different number of degrees of freedom and a different cost functional.

Moving on to the solution of dual variables, Fig. 4.10 shows the evolution of the mass costate along

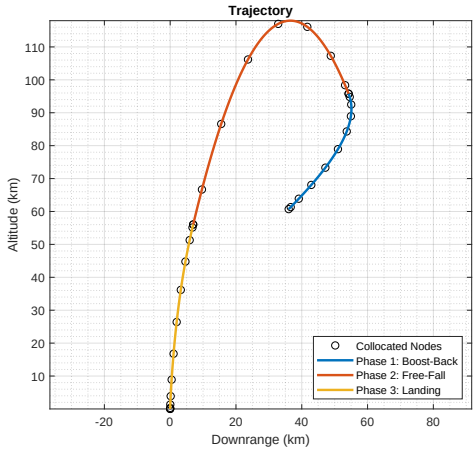


Figure 4.8: Altitude vs downrange trajectory. Boost-back phase is represented in blue; the coast phase is represented in orange; and the landing phase is shown in yellow.

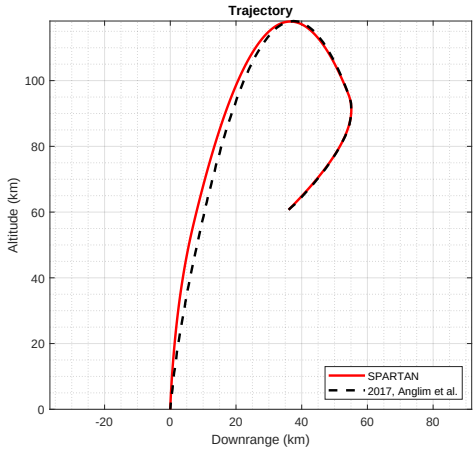


Figure 4.9: 2-D trajectory yielded by SPARTAN (in red) superimposed with the reference solution [67] (dashed black).

time. Recalling the first order necessary conditions from Section 2.3.1, it is possible to note that the endpoint condition expressed in (2.33) is verified at the final time of the trajectory, namely

$$\lambda_m(t_f) = \frac{\partial \Phi}{\partial m(t_f)} = -1\,. \tag{4.22}$$

This condition is not verified at any other phase endpoint due to the fact that the mass is either fixed at those points or completely determined by the initial conditions, the constant mass flow rates and the fixed endpoint time of phase 1. Ultimately, the validation of the endpoint condition at the final time of the trajectory brings confidence that the solution is optimal.

Another factor to consider in order to validate the optimality of the solution is whether the primer vector (velocity costate) is collinear with the thrust vector [5, 15]. Figure 4.11, therefore, shows the Cartesian components of this vector (on the left), and the norm of the cross product between this vector and the control (top-right and bottom-right plots). In order for the vectors to be collinear, it is required that the cross product between each other yields zero. It is clear that this collinearity is not achieved during the phase 1, in fact, the norm of the cross product is identical to the vertical component of the primer vector during this phase. This non-collinearity, however, is expected because the control is constrained during this phase of the trajectory, having no degrees of freedom available assert orientation concurrency with the primer vector. During phase 2, the throttle is constrained to be zero at all times, which means that the cross product yields a trivial null vector. However, during phase 3, the thrust is allowed to vary its direction and assert orientation concurrency. By looking at Figure 4.11 we see that the norm of the cross product is very small with respect to the components of both the primer vector and of the control, being roughly eight orders of magnitude smaller. With this information it is possible to assess that the two vectors are indeed collinear during phase 3. This result further increases confidence in the optimality of the solution.

Finally, Fig. 4.12 shows the Hamiltonian associated with the trajectory. By inspection of the plot, one can assert that the Hamiltonian is phase-wise constant, and with regards to phases 2 and 3, one can verify that the Hamiltonian is zero, which implies that the endpoint condition (2.35) is satisfied in these phases:

$$\mathcal{H}\left(t_f^{(2)}\right) = \mathcal{H}\left(t_f^{(3)}\right) = 0\,, \tag{4.23}$$

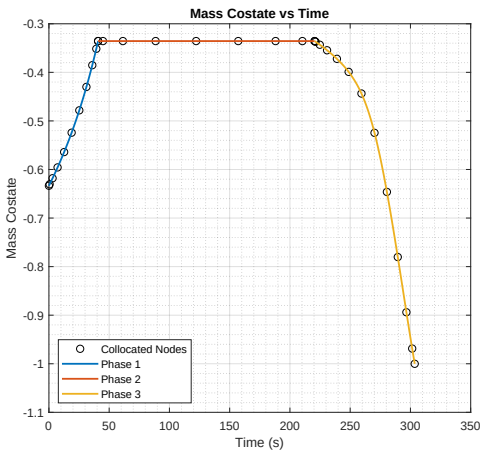


Figure 4.10: Mass costate with time. The mass costate takes the value -1 at the final time.

Briefly stated, the Hamiltonian satisfies the optimality conditions for time invariant systems [6, 9, 37], and therefore this result brings further confirmation that the solution is optimal.

Table 4.2 presents the values of pertinent variables obtained with SPARTAN. The results of the original article [67] are also presented for comparison. Briefly speaking, and disregarding the duration of the boost-back phase, the trajectory yielded by SPARTAN consists of a longer duration of coasting arc, and a shorter duration of powered descent when compared to the solution of the original article. This also results in a significantly higher final mass of the vehicle. The results are satisfactory overall.

Table 4.2: Comparison of relevant parameter results between SPARTAN and Anglim et al. [67].

Parameter	SPARTAN	Anglim et al. [67]	Unit
Boost-back burn duration, t_1	40.8	40.5684	s
Landing burn start time, t_2	220.4583	216.9112	s
Total time of flight, t_f	303.5469	307.8189	s
Final mass, $m(t_f)$	27905.5554	26221.1488	kg

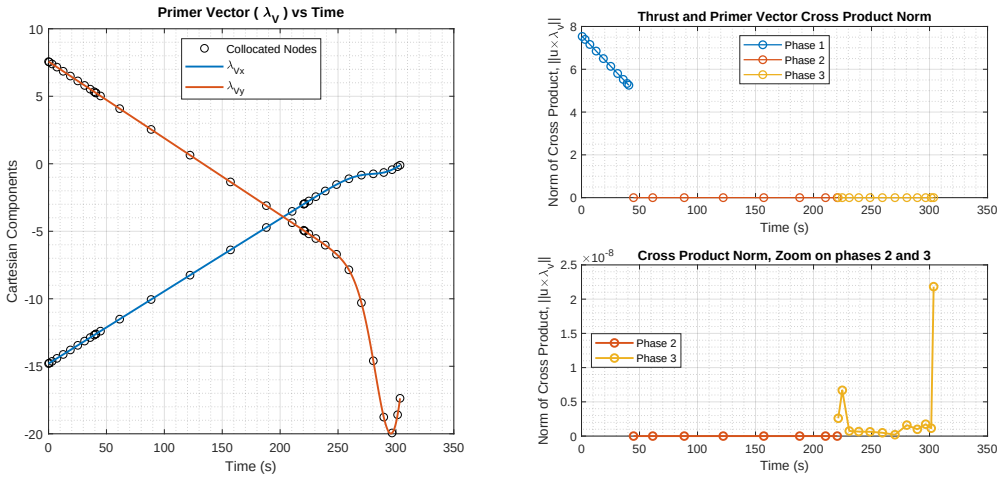


Figure 4.11: Left: Cartesian components of primer vector with time. Top-right: Norm of the cross product between the control and the primer vector. Bottom-Right: Zoom in on phases 2 and 3 of the top-right plot.

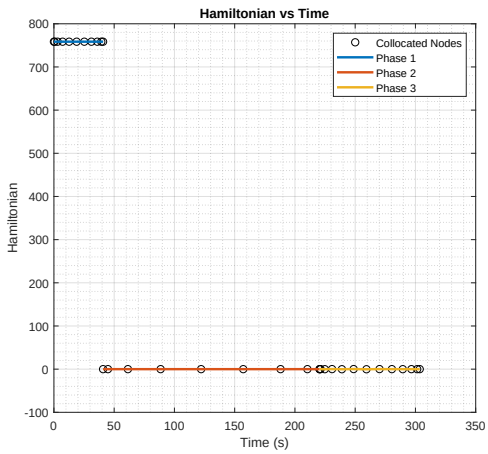


Figure 4.12: Hamiltonian vs time. Hamiltonian is zero during phases 2 and 3.

4.2 Problem 2: Delta III Rocket Ascent to Elliptical Orbit (4 phases, 3-D trajectory)

This example deals with the optimization of the ascent trajectory of a solid propellant multi-staged rocket taken from [7, 9, 10]. The vehicle associated with this problem is based on the characteristics of Boeing's Delta III rocket which is illustrated in Fig. 4.13. This problem is appropriate to validate the algorithm developed in this work because it is composed of four distinct phases. The modelling of the problem in four phases results from mutations in the equations of motion from one phase to the next which yield removable discontinuities on the velocity, but also from jump discontinuities in the mass of the vehicle due to stage separation events. The algorithm is validated by analysis of the optimality conditions and also by comparison of the results with the reference solution.

This problem occurs in three dimensional space (3-D) and it is about finding the trajectory from launch pad to orbit insertion of a vehicle composed of nine solid rocket boosters, one main stage, and one upper stage. In total there are four distinct phases, $P = 4$. The flight sequence is:

Phase 1 Ignition of the main stage and of six solid boosters (out of nine) at $t = t_0$. Full throttle until the burnout of the six solid boosters. The six empty boosters are separated and dropped out at $t = t_1$.

Phase 2 Main stage continues in full throttle during all of phase 2. Ignition of the three left over boosters at $t = t_1$. Full throttle until the burnout of the boosters. Three empty boosters are separated and dropped out at $t = t_2$.

Phase 3 Main stage continues in full throttle. Burn out of the remaining fuel in the main stage. Main stage is separated and dropped out at $t = t_3$.

Phase 4 Ignition of the upper stage at $t = t_3$. Full throttle until orbit insertion. Payload deployed at $t = t_4$.

The equations of motion are taken directly form [9]. Figure 4.14 presents the free-body diagram of the vehicle for visual reference. The position vector is represented by $\boldsymbol{r}$, the velocity vector is $\boldsymbol{v}$, and the aerodynamic drag force is represented by $\boldsymbol{D}$. The thrust vector, is shown as $\boldsymbol{T}$. The air is assumed to be "attached" to the Earth, sharing its rotation. The acceleration of gravity, $\boldsymbol{g}$, is collinear with the position vector $\boldsymbol{r}$; the aerodynamic drag force, $\boldsymbol{D}$, is collinear with the spacecraft's velocity relative to local air, $\boldsymbol{v}_{\text{rel}}$, and the

Figure 4.13: Delta III rocket illustration [69].

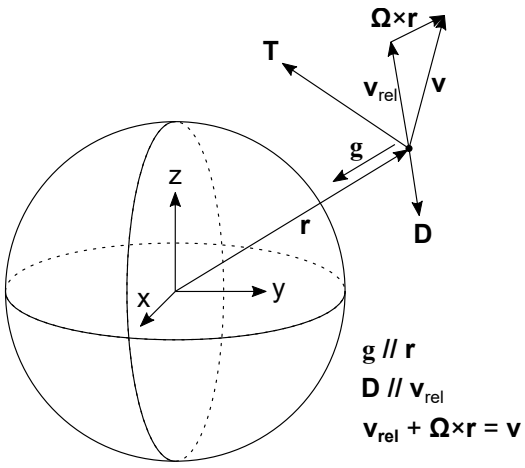


Figure 4.14: Free-body diagram of the vehicle in ECI coordinates. The Earth is represented by a sphere centred at the origin.

inertial velocity of the spacecraft, $\boldsymbol{v}$, can be decomposed into velocity relative to local air, $\boldsymbol{v}_{\text{rel}}$, and inertial air velocity, $\boldsymbol{\Omega} \times \boldsymbol{r}$, where $\boldsymbol{\Omega}$ is the Earth's angular velocity vector relative to the inertial reference frame. The Earth is assumed to be spherical and rotating about the z axis, and the acceleration of gravity decays with the radial distance to the centre of the planet as [9]

$$\boldsymbol{g} = -\frac{\mu}{\|\boldsymbol{r}\|^3}\boldsymbol{r}\,, \tag{4.24}$$

where μ is the standard gravitational parameter of the Earth. The mass of the vehicle is represented by m, and the control variable is the thrust unit vector, $\boldsymbol{u}$. A point-mass approximation of the vehicle is employed, thus the attitude of the spacecraft is not modelled and the angle of attack is assumed to be zero at all times. An exponential model is used for the drag force. The optimal control problem is stated as minimizing the cost functional

$$\mathcal{J} = \Phi\big(m(t_f)\big) = -m(t_f)\,, \tag{4.25}$$

subject to the equations of motion

$$\dot{\boldsymbol{r}} = \boldsymbol{v}\,, \tag{4.26}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{g} + \frac{T}{m}\boldsymbol{u} + \frac{\boldsymbol{D}}{m}\,, \tag{4.27}$$

$$\dot{m} = -\frac{T}{g_0 I_{sp}}\,, \tag{4.28}$$

the initial conditions

$$\boldsymbol{r}(t_0^{(1)}) = \begin{bmatrix} 5605.2 \times 10^3 & 0 & 3043.4 \times 10^3 \end{bmatrix}^{\mathsf{T}} \text{m}\,, \tag{4.29}$$

$$\boldsymbol{v}(t_0^{(1)}) = \begin{bmatrix} 0 & 0.4076 \times 10^3 & 0 \end{bmatrix}^{\mathsf{T}} \text{m}\,, \tag{4.30}$$

$$m(t_0^{(1)}) = 301\,454\,\text{kg}\,, \tag{4.31}$$

the terminal conditions (orbital elements)

$$\text{semi-major axis} \quad a_f^{(4)} = 24\,361.14\,\text{km}\,, \tag{4.32}$$

$$\text{eccentricity} \quad e_f^{(4)} = 0.7308\,, \tag{4.33}$$

$$\text{inclination} \quad i_f^{(4)} = 28.5°, \tag{4.34}$$

$$\text{longitude of ascending node} \quad \Omega_f^{(4)} = 269.8°, \tag{4.35}$$

$$\text{argument of perigee} \quad \omega_f^{(4)} = 130.5°, \tag{4.36}$$

$$\text{true anomaly} \quad \nu_f^{(4)} = \text{free}\,, \tag{4.37}$$

and the phase linkage conditions

$$\boldsymbol{r}(t_0^{(p+1)}) - \boldsymbol{r}(t_f^{(p)}) = \boldsymbol{0}\,, \tag{4.38}$$

$$\boldsymbol{v}(t_0^{(p+1)}) - \boldsymbol{v}(t_f^{(p)}) = \boldsymbol{0}\,, \tag{4.39}$$

$$m(t_0^{(p+1)}) - m(t_f^{(p)}) = -m_{\text{dry}}^{(p)}\,. \tag{4.40}$$

Where T is the thrust modulus (constant along each phase), g_0 is the Earth's gravity acceleration, I_{sp} is the specific impulse (constant along each phase) and D is the aerodynamic drag force. The drag force assumes an exponential model of the atmospheric air density and it is computed as

$$\boldsymbol{D} = -\frac{1}{2}\rho S C_D \|\boldsymbol{v}_{\text{rel}}\| \boldsymbol{v}_{\text{rel}}\,, \tag{4.41}$$

with

$$\rho = \rho_0 \exp\{-h/h_0\}\,, \tag{4.42}$$

$$h = \|\boldsymbol{r}\| - R_e\,, \tag{4.43}$$

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & 0 & \Omega_\oplus \end{bmatrix}^\intercal\,, \tag{4.44}$$

$$\boldsymbol{v}_{\text{rel}} = \boldsymbol{v} - \boldsymbol{\Omega} \times \boldsymbol{r}\,. \tag{4.45}$$

Where S is the reference surface area, C_D is the drag coefficient, ρ is the air density as a function of altitude, ρ_0 is the standard air density at sea level, h is the altitude of the spacecraft, h_0 is the density scale height, R_e is the radius of the Earth and $\Omega_\oplus$ is the angular velocity of Earth's rotation. Tables 4.3 and 4.4 show the values of all the constants and vehicle properties mentioned above.

It is relevant to note that the intermediate phase transition times t_1, t_2 and t_3 are known (fixed), but the final time of phase 4 is open and to be determined. The problem was solved with the collocation of 8, 8, 8, and 16 flipped Radau nodes on phases 1, 2, 3 and 4, respectively. The NLP solver used was IPOPT [28], and the computation of partial derivatives was done through the complex step differentiation method [48, 49].

The results for this problem are presented in Figs. 4.15 through 4.23. More specifically, Figs. 4.15 through 4.20 deal with the solution of state and control, while Figs. 4.22 through 4.23 are related to the dual variables and the optimality of the solution.

Figures 4.15 and 4.16 show the altitude of the spacecraft and the norm of the velocity vector with time, respectively. It can be seen that the altitude, expressed with respect to the Earth radius, starts at zero with a path that is tangential to the time axis, and goes to about 200 km where orbit insertion occurs. The profile of the altitude seems to "wobble" during the last phase where it decreases after reaching a local maximum, only

Table 4.3: Relevant constants and parameters for the multistage solid propellant rocket problem [10].

Constant	Value	Unit
Payload mass, m_{payload}	4164	kg
Reference surface area, S	4π	m^2
Drag Coefficient, C_D	0.5	1
Air density at sea level, ρ_0	1.225	$\text{kg}\,\text{m}^{-3}$
Density scale height, h_0	7200	m
Earth radius, R_e	6378145	m
Earth rotation rate, $\Omega_\oplus$	$7.292\,115\,85 \times 10^{-5}$	$\text{rad}\,\text{s}^{-1}$
Standard gravitational parameter, μ	$3.986\,012 \times 10^{14}$	$\text{m}^3\,\text{s}^{-2}$
Gravity acceleration at sea level, g_0	9.80665	$\text{m}\,\text{s}^{-2}$
Phase 1 initial time, t_0	0	s
Phase 1 to phase 2 transition time, t_1	75.2	s
Phase 2 to phase 3 transition time, t_2	150.4	s
Phase 3 to phase 4 transition time, t_3	261	s

Table 4.4: Component properties of the vehicle for the multistage solid propellant rocket problem [9].

	Solid Fuel Boosters	Main Stage	Upper Stage	Unit
Total Mass, $m_{\text{component}}$	19290	104380	19300	kg
Propellant Mass, m_{prop}	17010	95550	16820	kg
Dry Mass, $m_{\text{dry}} = m_{\text{component}} - m_{\text{prop}}$	2280	8830	2480	kg
Engine Thrust, T	628500	1083100	110094	N
Burn Time, t_b	75.2	261	700	s
Number of Engines	9	1	1	1
Specific Impulse, I_{sp}		$T\,t_b/(g_0\,m_{\text{prop}})$		s

to increase again for the orbit insertion point. The velocity norm starts a little above zero due to the tangential component introduced by the Earth's rotation at the surface and it follows a non-decreasing profile along the trajectory.

Figures 4.17 and 4.18 present a direct comparison between the results obtained with SPARTAN and the reference solution [9]. The figures show the same variables depicted in Fig. 4.15 and Fig. 4.16, respectively: altitude and norm of velocity. The two solutions are superimposed for direct comparison with the state-of-the-art solver GPOPS [9]. With regards to both plots, it can be seen that the two solutions show excellent agreement with each other, being virtually indistinguishable.

The profile of the total mass of the vehicle is shown in Fig. 4.19. In agreement with the equations of motion, the mass decays linearly due to constant mass flow rate on each phase (steeper mass gradients correspond to larger mass flow rates). Also, it is possible to see that the plot presents clear jump discontinuities at the phase transitions, this is the direct result of stage separation events where empty rocket boosters are discarded. Jump discontinuities in the mass of the system represent a main motivation for the employment of a multiphase algorithm. This characteristic of the problem makes it adequate to validate the flipped Radau pseudospectral method.

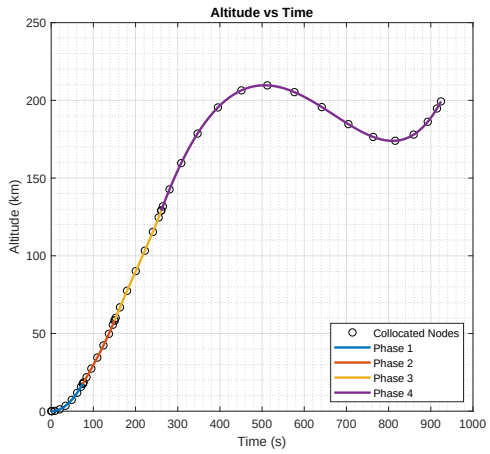


Figure 4.15: Spacecraft altitude vs time.

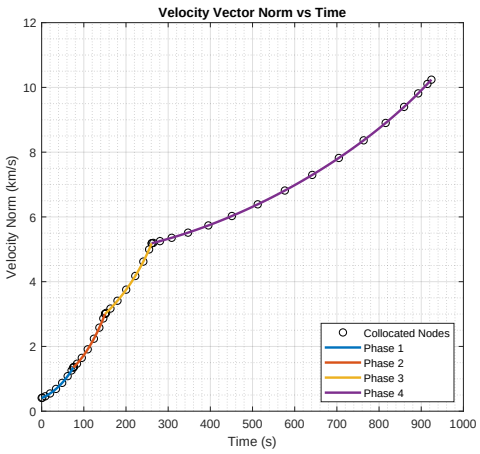


Figure 4.16: Norm of the velocity vector vs time.

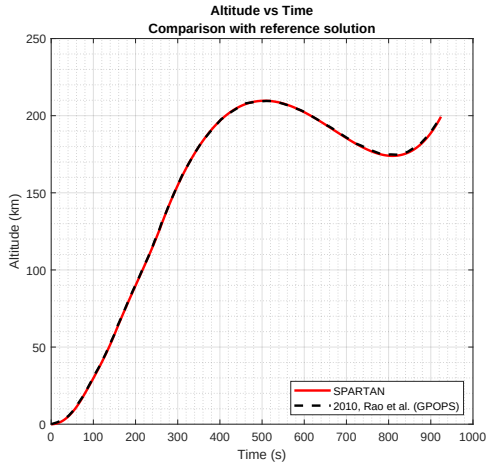


Figure 4.17: Spacecraft altitude vs time. Comparison of results with the reference solution [9].

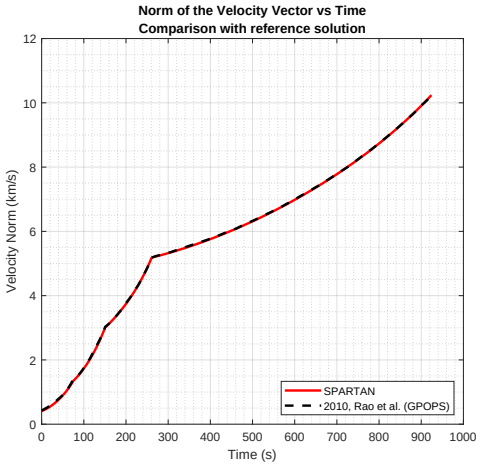


Figure 4.18: Velocity norm vs time. Comparison of results with the reference solution [9].

The Cartesian components of the control, $\boldsymbol{u}$, are presented in Fig. 4.20. In this case the phase transition points are evidenced by the increased density of collocated nodes along the time domain. It can be asserted that the components assert a unit vector throughout the trajectory. The representation of thrust in unit vector form is convenient to preserve continuity, as the modulus of thrust changes instantaneously at every phase (the thrust is subject to jump discontinuities). The smoothness of the thrust unit vector along the trajectory indicates that the thrust vector, although subject to jump discontinuities in its norm, preserves smoothness in its direction regardless of the phase transitions.

With regards to Fig. 4.21 there a few relevant points to mention. The left plot presents the mass costate in all 4 phases, while the plot on the right shows a zoom in version of the mass costate that focuses on phases 1, 2 and 3. By inspection of both illustrations in this Figure one can extract the following information:

$$\boldsymbol{\lambda}_m(t_0^{(2)}) = 0\,, \tag{4.46}$$

$$\boldsymbol{\lambda}_m(t_0^{(3)}) = 0\,, \tag{4.47}$$

$$\boldsymbol{\lambda}_m(t_f^{(4)}) = -1\,. \tag{4.48}$$

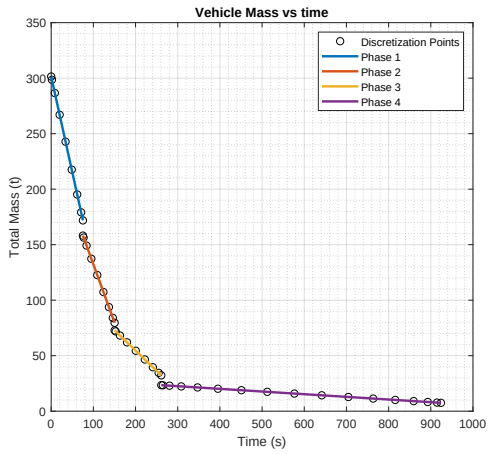


Figure 4.19: Decay in total vehicle mass along the trajectory.

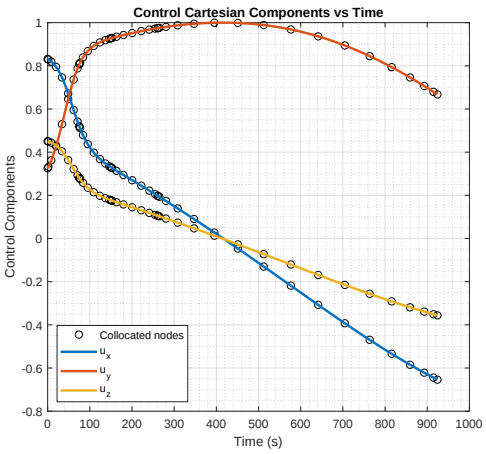


Figure 4.20: Cartesian components of control. The components assert a unit vector.

These equalities indicate the validity of the endpoint conditions expressed in (2.32) and (2.33) from Chapter 2. Namely, equation (2.32) applies in phases 2 and 3, and equation (2.33) applies in phase 4. Neither of these two conditions apply in phase 1. This is because the initial mass of phase 1 is constrained to a fixed value, $\delta m_0^{(1)} = 0$, and thus, the exclusive disjunction property applies. No further conditions apply to the endpoints of phase 1 because the mass has a constant flow-rate: a set initial value and a constant flow-rate imply a set final value in the case of fixed final time. The exclusive disjunction property applies also at the final endpoint of phase 1, $\delta m_f^{(1)} = 0$. With regards to phase 4, there is also an explicit constraint on the initial mass to a fixed value, $\delta m_0^{(4)} = 0$, and the mass flow-rate is also constant, which is a similar scenario to phase 1, however, phase 4 consists of an unconstrained final time, $\delta t_f^{(4)} \neq 0$, and this means that the final mass is also unconstrained $\delta m_f^{(4)} \neq 0$ (although completely determined once a final time is obtained). This leaves one degree of freedom available to apply the endpoint condition in (2.33). Curiously, with regards to phases 2 and 3, either (2.32) or (2.33) (but not both) could be employed, as the mass is unconstrained on both endpoints of these phases. Presumably the NLP solver "chooses" to employ the one which will minimize the Hamiltonian given an initial guess of costates, even though this decision will not affect the solution of state and control. Ultimately, the mass costate satisfies the necessary conditions described in Chapter 2, which is indicative of an optimal solution.

Further, Fig. 4.22 presents two pertinent plots. The plot on the left illustrates the Cartesian components of the velocity costate (or primer vector), and the plot on the right shows the norm of the cross product between this vector and the thrust unit vector (control), shown in Fig. 4.20. It is known that an optimal trajectory requires the thrust vector to be collinear with the primer vector at all times (provided that the thrust is not constrained) [5, 15]. With this in mind, and by looking at the right plot of Fig. 4.22, one can notice that the norm of the cross product between thrust and primer vector is very close to zero throughout the trajectory, indicating that these two vectors are very close to being collinear. This collinearity brings further evidence to prove the optimality of the solution.

It is well known that every optimal control problem can be formulated in multiple ways without changing what will be the optimal solution of state and control, but the same cannot be said about the Hamiltonian

[6]. The solution of the Hamiltonian is not unique to a given trajectory of state and control. For instance, in this particular case, the mass flow-rate is constant on every phase $\dot{m} = \text{Constant}$, and therefore the problem could be formulated by having open intermediate times, t_1, t_2 and t_3, and fixed endpoint masses (instead of fixed times and open masses), in which case the endpoint condition in (2.35) would apply at the terminal point of each phase, forcing the Hamiltonian to be zero everywhere. Nevertheless, the solution of state and control would be identical to the present one, as no additional degrees of freedom would be introduced.

By looking at Fig. 4.23, it is clear that the Hamiltonian yielded by SPARTAN does not concur with the reference solution [9] in its entirety. The divergence of the Hamiltonian in phases 1 and 3, is thus, most likely, due to a slight difference in formulations between the two approaches. Finding the exact formulation which yields the Hamiltonian of the original source is a "reverse-engineering" problem that goes beyond the scope of this thesis. As discussed above, although different formulations might affect the Hamiltonian solution, they do not affect the solution of state and control, provided the degrees of freedom remain the same. In other words, the divergence of the Hamiltonian is not a relevant factor to determine optimality in this case. A closer look

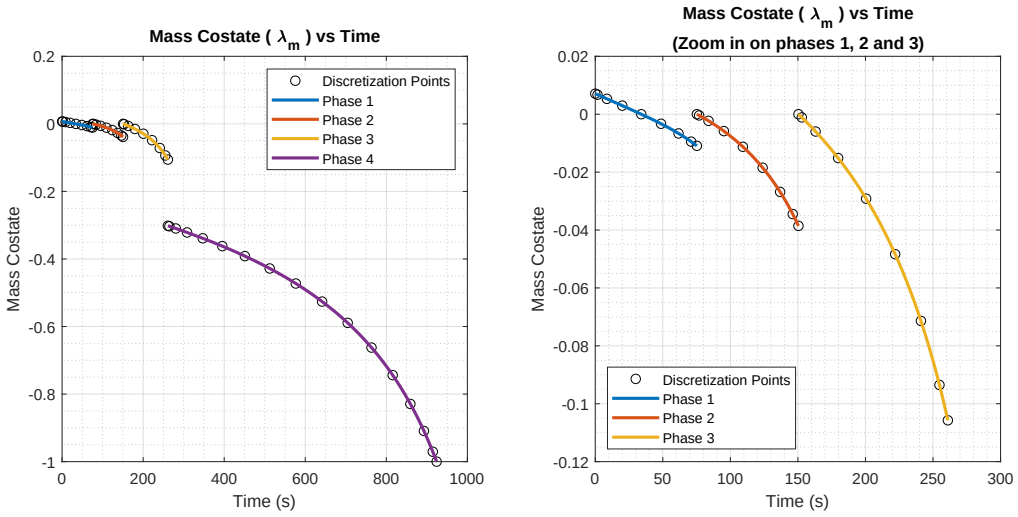


Figure 4.21: Left: Mass costate along the trajectory. Right: Zoom in on phases 1, 2 and 3 of the left plot.

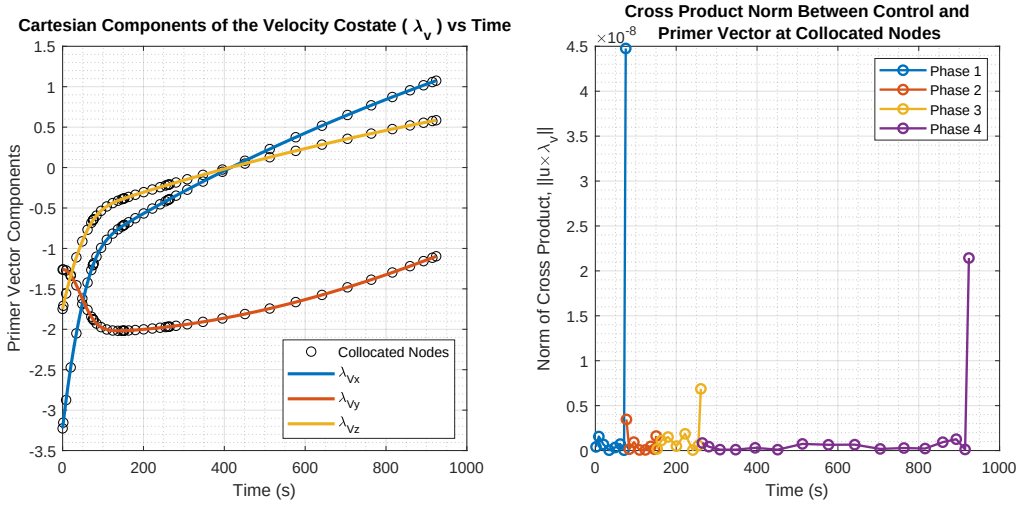


Figure 4.22: Left: Cartesian components of the velocity costate (primer vector) at each time instant. Right: Norm of the cross product between the control and the primer vector.

at Fig. 4.23 reveals something that is verified in both results, namely, the validation of the endpoint condition from (2.35) in phase 4,

$$\mathcal{H}(t_f^{(4)}) = 0\,. \tag{4.49}$$

This condition is valid because the final time of phase 4 is open, $\delta t_f^{(4)} \neq 0$. So, because the optimal Hamiltonian is constant [9, 37], then it must also be zero everywhere along this phase. Notice that neither (2.34) or (2.35) apply to phases 1, 2 or 3, and this is because the endpoint times of these phases are fixed (exclusive disjunction property). Ultimately, Fig. 4.23 shows that SPARTAN yields a phase-wise constant Hamiltonian, implying that the solution is indeed optimal.

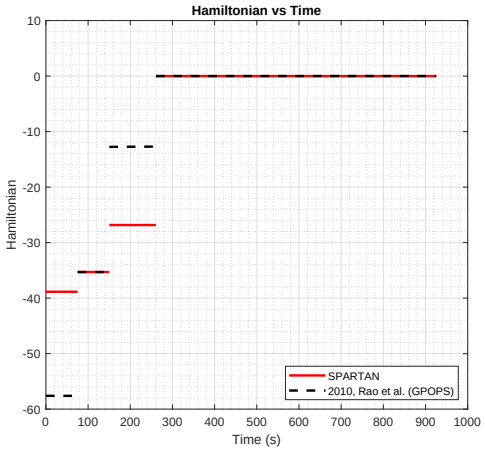


Figure 4.23: Comparison between the Hamiltonian obtained with SPARTAN and the reference solution [9].

Table 4.5 presents the results of relevant parameters associated with the problem, namely the final time of phase 4 and the final mass of the vehicle (Mayer cost). The results from [9] are also presented. In can be noted that the final mass of the vehicle is only a couple hundred grams higher in the solution of SPARTAN with regards to the solution of the original article. This increases confidence that the trajectories are virtually identical. Although the final time is unspecified in the original article, the results should be very similar as well due to the identical initial masses at the beginning of phase 4 and the constant mass flow rate during this phase. Ultimately, the results yielded by SPARTAN with the flipped Radau method are satisfactory.

Table 4.5: Comparison of relevant parameter results between SPARTAN and Rao et al. [9].

Parameter	SPARTAN	Rao et al. [9]	Unit
Total time of flight, t_f	924.13043	unspecified	s
Final mass, $m(t_f)$	7529.9284	7529.7123	kg

Chapter 5

Conclusions

In this work the flipped Radau pseudospectral method was applied to solve multiphase optimal control problems, specifically multi-stage rocket trajectory generation problems in both ascension to orbit and descent to vertical landing. The algorithm was validated by solving two relevant reference problems containing multiple phases each.

A simplified version of the first reference problem was implemented, which concerned a booster recovery of the Falcon 9 orbital launcher. Despite the difference in the formulations, the results were satisfactory as the solution was shown to be optimal by analysis of the dual variables and Hamiltonian. Also, by direct comparison with the reference solution, the trajectories diverged only slightly, making the simplified problem scenario plausible by the standards of the original article.

With respect to the second example, concerning the ascent and orbit insertion of the Delta III rocket, not only were the optimality conditions verified by the dual variable analysis, but the results showed excellent agreement with the reference solution. Ultimately, the implementation of the flipped Radau pseudospectral method in multiphase problems can be deemed successful, and further, it can be assessed that the accuracy obtained is comparable to state of the art solvers.

Regarding future developments in booster recovery trajectory generation, given that the era of reusable rockets has arrived, the implementation of an algorithm capable of dealing with non-sequential phases could be an interesting research topic, as it would allow the optimization of both the ascending path and the booster recovery path simultaneously.

References

[1] G. P. Sutton and O. Biblarz. *Rocket Propulsion Elements*. John Wiley & Sons Inc, ninth edition, 2016. ISBN 1118753658.

[2] Isakowitz, Steven. *International Reference Guide to Space Launch Systems*. American Institute of Aeronautics and Astronautics, Reston, Va, 2004. ISBN 156347591X.

[3] W. Wiesel. *Spaceflight dynamics*. Aphelion Press, Beavercreek, Ohio, 2010. ISBN 9781452879598.

[4] T. S. Taylor. *Introduction to Rocket Science and Engineering*. Taylor & Francis Inc, 2017. ISBN 1498772323.

[5] N. X. Vinh. General Theory of Optimal Trajectory for Rocket Flight in a Resisting Medium. *Journal of Optimization Theory and Applications*, 11(2):189–202, feb 1973. doi: 10.1007/bf00935883.

[6] Arthur E. Bryson Jr. and Yu-Chi Ho. *Applied Optimal Control. Optimization, Estimation, and Control*. Hemisphere Publishing Corporation, Washington New York, 1975. ISBN 9780891162285.

[7] J. T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. SIAM, Philadelphia, second edition, 2010. ISBN 978-0-898716-88-7.

[8] D. E. Kirk. *Optimal Control Theory, An Introduction*. Dover Publications Inc., 2004. ISBN 0486434842.

[9] A. V. Rao, D. A. Benson, C. L. Darby, M. A. Patterson, C. Francolin, I. Sanders, and G. T. Huntington. Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method. *ACM Transactions on Mathematical Software*, 37(2):1–39, Apr. 2010. doi: 10.1145/1731022.1731032.

[10] M. A. Patterson and A. V. Rao. GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming. *ACM Transactions on Mathematical Software*, 41(1):1–37, October 2014. doi: 10.1145/2558904.

[11] Naidu, D. S. *Optimal Control Systems*. CRC Press, Boca Raton, Fla, 2003. ISBN 0849308925.

[12] F. Lewis. *Optimal Control*. John Wiley & Sons, Hoboken, 2012. ISBN 9781118122648.

[13] D. F. Lawden. *Optimal Trajectories for Space Navigation*. Butterworths, London, 1963.

[14] J. T. Betts. Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, mar 1998. doi: 10.2514/2.4231.

[15] B. Conway. *Spacecraft Trajectory Optimization*. Cambridge University Press, Cambridge New York, 2010. ISBN 9780511909450.

[16] H. J. Sussmann and J. C. Willems. 300 Years of Optimal Control: from the Brachystochrone to the Maximum Principle. *IEEE Control Systems*, 17(3):32–44, June 1997. doi: 10.1109/37.588098.

[17] L. S. Pontryagin and V. G. Boltyanskii and R. V. Gamkrelidze and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. Interscience Publishers, John Wiley & Sons, 1962.

[18] Michael Athans and Peter L. Falb. *Optimal Control. An Introduction to the Theory and Its Applications*. Dover Publications Inc., 2006. ISBN 0486453286.

[19] D. A. Benson, G. T. Huntington, T. P. Thorvaldsen, and A. V. Rao. Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method. *Engineering Notes, Journal of Guidance, Control, and Dynamics*, 29(6):1435–1440, Nov. 2006. doi: 10.2514/1.20478.

[20] G. Elnagar, M. A. Kazemi, and M. Razzaghi. The pseudospectral Legendre method for discretizing optimal control problems. *IEEE Transactions on Automatic Control*, 40(10):1793–1796, 1995. doi: 10.1109/9.467672.

[21] F. Fahroo and I. M. Ross. Costate Estimation by a Legendre Pseudospectral Method. *Journal of Guidance, Control, and Dynamics*, 24(2):270–277, mar 2001. doi: 10.2514/2.4709.

[22] F. Fahroo and I. M. Ross. Direct Trajectory Optimization by a Chebyshev Pseudospectral Method. *Journal of Guidance, Control, and Dynamics*, 25(1):160–166, Jan. 2002. ISSN 0731-5090. doi: 10.2514/2.4862.

[23] I. M. Ross and F. Fahroo. Legendre Pseudospectral Approximations of Optimal Control Problems. *Springer*, 295:327–342, 2003. doi: 10.1007/978-3-540-45056-6_21.

[24] Q. Gong, W. Kang, N. S. Bedrossian, F. Fahroo, P. Sekhavat, and K. Bollino. Pseudospectral Optimal Control for Military and Industrial Applications. In *2007 46th IEEE Conference on Decision and Control*, pages 4128–4142. IEEE, 2007. doi: 10.1109/cdc.2007.4435052.

[25] Fornberg, Bengt. *A practical guide to pseudospectral methods*. Cambridge University Press, Cambridge New York, 1996. ISBN 0521495822.

[26] P. E. Gill, E. Wong, W. Murray, and M. A. Saunders. User's Guide for SNOPT Version 7.6: Software for Large-Scale Nonlinear Programming. *University of California, San Diego*, January 2017.

[27] A. Wächter and L. Biegler. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical programming*, 106:25–57, March 2005. doi: 10.1007/s10107-004-0559-y.

[28] Y. Kawajir, C. Laird, S. Vigerske, and A. Wächter. Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT. *Chicago Northwestern University*, April 2015.

[29] Fariba Fahroo and I. Michael Ross. Advances in Pseudospectral Methods for Optimal Control. In *AIAA Guidance, Navigation, and Control Conference and Exhibit, Honolulu, USA, 2008*, pages 1–23, 2008. doi: 10.2514/6.2008-7309.

[30] I. M. Ross. A Historical Introducion to the Covector Mapping Principle. In *AAS / AIAA Astrodynamics Specialist Conference, Tahoe, NV, USA*, pages 1–21, 2005.

[31] Q. Gong, I. M. Ross, W. Kang, and F. Fahroo. Connections Between The Covector Mapping Theorem and Convergence of Pseudospectral Methods for Optimal Control. *Computational Optimization and Applications*, 41(3):307–335, oct 2008. doi: 10.1007/s10589-007-9102-4.

[32] Marco Sagliano and Stephan Theil and Vincenzo D'Onofrio and Michiel Bergsma. SPARTAN: A Novel Pseudospectral Algorithm for Entry, Descent, and Landing Analysis. In *Advances in Aerospace Guidance, Navigation and Control*, pages 669–688. Springer International Publishing, dec 2017. doi: 10.1007/978-3-319-65283-2_36.

[33] L. Trefethen. *Spectral methods in MATLAB*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000. ISBN 9780898714654.

[34] G. T. Huntington and A. V. Rao. Comparison of Global and Local Collocation Methods for Optimal Control. *Journal of Guidance, Control, and Dynamics*, 31(2):432–436, Mar. 2008. ISSN 0731-5090. doi: 10.2514/1.30915.

[35] D. A. Benson. *A Gauss Pseudospectral Transcription for Optimal Control*. PhD thesis, Massachusettes Institute of Technology, 2004.

[36] I. M. Ross, P. Sekhavat, A. Fleming, Q. Gong, and W. Kang. Pseudospectral Feedback Control: Foundations, Examples and Experimental Results. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–23. American Institute of Aeronautics and Astronautics, Aug. 2006. doi: 10.2514/6.2006-6354.

[37] D. Garg. *Advances in Global Pseudospectral Methods for Optimal Control*. PhD thesis, University of Florida, 2011.

[38] K. P. Bollino. *High-Fidelity Real-Time Trajectory Optimization for Reusable Launch Vehicles*. PhD thesis, Naval Postgraduate School, Monterey, 2006.

[39] V. M. Becerra. Solving complex optimal control problems at no cost with PSOPT. In *2010 IEEE International Symposium on Computer-Aided Control System Design*, pages 1391–1396. IEEE, sep 2010. doi: 10.1109/cacsd.2010.5612676.

[40] Y. Nie, O. Faqir, and E. C. Kerrigan. ICLOCS2: Try this Optimal Control Problem Solver Before you Try the Rest. In *2018 UKACC 12th International Conference on Control (CONTROL)*. IEEE, sep 2018. doi: 10.1109/control.2018.8516795.

[41] M. Sagliano and S. Theil. Hybrid Jacobian Computation for Fast Optimal Trajectories Generation. In *AIAA Guidance, Navigation, and Control Conference, Boston, USA*, 2013. doi: 10.2514/6.2013-4554.

[42] L. Huneker and M. Sagliano and Y. E. Arslantaş. SPARTAN: An Improved Global Pseudospectral Algorithm for High-Fidelity Entry-Descent-Landing Guidance Analysis. In *30th International Symposium on Space Technology and Science, Kobe, Japan, 2015*, 2015.

[43] M. Sagliano. *Development of a Novel Algorithm for High Performance Reentry Guidance*. PhD thesis, Universität Bremen, 2016.

[44] M. Sagliano, E. Mooij, and S. Theil. Onboard trajectory generation for entry vehicle via adaptive multivariate pseudospectral interpolation. In *AIAA Science and Technology Forum and Exposition*, 2016.

[45] Yunus Emre Arslantaş and Thimo Oehlschlägel and Marco Sagliano. Safe landing area determination for a Moon lander by reachability analysis. *Acta Astronautica*, 128:607–615, nov 2016. ISSN 0094-5765. doi: 10.1016/j.actaastro.2016.08.013.

[46] Marco Sagliano. Pseudospectral Convex Optimization for Powered Descent and Landing. *Journal of Guidance, Control and Dynamics*, 41(2), 2018. doi: 10.2514/1.G002818.

[47] N. S. Bedrossian and S. Bhatt and W. Kang and I. M. Ross. Zero-propellant maneuver guidance. *IEEE Control Systems Magazine*, 29(5):53–73, Oct. 2009. ISSN 1941-000X. doi: 10.1109/MCS.2009.934089.

[48] J. R. R. Martins, P. Sturdza, and J. J. Alonso. The Complex-Step Derivative Approximation. *ACM Transactions on Mathematical Software, Vol. 29, No. 3, September 2003, Pages 245-262*, 2003. doi: 10.1145/838250.838251.

[49] V. D'Onofrio. Implementation of Advanced Differentiation Methods for Optimal Trajectory Computation. Master's thesis, Università Degli Studi Di Napoli "Federico II", 2014.

[50] D'Onofrio, Vincenzo and Sagliano, Marco and Arslantaş, Yunus E. Exact Hybrid Jacobian Computation for Optimal Trajectory Generation via Dual Number Theory. In *AIAA Guidance, Navigation, and Control Conference*. American Institute of Aeronautics and Astronautics, January 2016. doi: 10.2514/6.2016-0867.

[51] M. Sagliano, S. Theil, M. Bergsma, V. D'Onofrio, L. Whittle, and G. Viavattene. On the Radau Pseudospectral Method: theoretical and implementation advances. *CEAS Space Journal*, 9(3):313–331, June 2017. doi: 10.1007/s12567-017-0165-5.

[52] Y. M. Agamawi, W. W. Hager, and A. V. Rao. Mesh Refinement Method for Solving Bang-Bang Optimal Control Problems Using Direct Collocation. *AIAA Scitech 2020 Forum*, Jan 2020. doi: 10.2514/6.2020-0378.

[53] A. V. Rao. A Survey of Numerical Methods for Optimal Control. In *AAS/AIAA Astrodynamics Specialist Conference, AAS Paper 09-334, Pittsburgh, PA, August 10 - 13*, pages 1–32, 2009.

[54] M. Sagliano. Performance analysis of linear and nonlinear techniques for automatic scaling of discretized control problems. *Operations Research Letters, Vol.42 Issue 3, May 2014, pp. 213-216*, 2014. doi: 10.1016/j.orl.2014.03.003.

[55] I. M. Ross, Q. Gong, M. Karpenko, and R. J. Proulx. Scaling and Balancing for High-Performance Computation of Optimal Controls. *Journal of Guidance, Control, and Dynamics*, 41(10):2086–2097, oct 2018. doi: 10.2514/1.g003382.

[56] D. Garg, M. Patterson, W. W. Hager, A. V. Rao, D. A. Benson, and G. T. Huntington. A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica*, 46 (11):1843–1851, nov 2010. doi: 10.1016/j.automatica.2010.06.048.

[57] D. Garg, W. W. Hager, and A. V. Rao. Pseudospectral methods for solving infinite-horizon optimal control problems. *Automatica*, 47(4):829–837, apr 2011. doi: 10.1016/j.automatica.2011.01.085.

[58] C. L. Darby and A. V. Rao. A Mesh Refinement Algorithm for Solving Optimal Control Problems Using Pseudospectral Methods. *American Institude of Aeronautics and Astronautics*, 2009.

[59] M. Sagliano. Generalized hp Pseudospectral-Convex Programming for Powered Descent and Landing. *Journal of Guidance, Control, and Dynamics*, 42(7):1562–1570, jul 2019. doi: 10.2514/1.g003731.

[60] N. Koeppen, I. M. Ross, L. C. Wilcox, and R. J. Proulx. Fast Mesh Refinement in Pseudospectral Optimal Control. *Journal of Guidance, Control, and Dynamics*, 42(4):711–722, 2019. doi: 10.2514/1.G003904.

[61] Q. Gong, F. Fahroo, and I. M. Ross. Spectral Algorithm for Pseudospectral Methods in Optimal Control. *Journal of Guidance, Control, and Dynamics*, 31(3):460–471, may 2008. doi: 10.2514/1.32908.

[62] J. D. Eide, W. W. Hager, and A. V. Rao. Modified Radau Collocation method for Solving Optimal Control Problems with Nonsmooth Solutions Part I: Lavrentiev Phenomenon and the Search Space. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, dec 2018. doi: 10.1109/cdc.2018.8619830.

[63] J. D. Eide, W. W. Hager, and A. V. Rao. Modified Radau Collocation Method for Solving Optimal Control Problems with Nonsmooth Solutions Part II: Costate Estimation and the Transformed Adjoint System. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, dec 2018. doi: 10.1109/cdc.2018.8619426.

[64] R. Radau. Étude sur les formules d'approximation qui servent à calculer la valeur numérique d'une intégrale définie. *Journal de mathématiques pures et appliquées*, 6:283–336, 1880.

[65] P. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Academic Press, Orlando, 1984. ISBN 9780122063602.

[66] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange Interpolation. *SIAM Review*, 46(3):501–517, jan 2004. doi: 10.1137/s0036144502417715.

[67] K. S. G. Anglim, Z. Zhang, and Q. Gao. Minimum-Fuel Optimal Trajectory For Reusable First-Stage Rocket Landing Using Particle Swarm Optimization. *World Academy of Science, Engineering and Technology, International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, 11(5):981–990, 2017. doi: 10.5281/ZENODO.1130268.

[68] SpaceX. SpaceX - Falcon 9, 2020. URL https://www.spacex.com/vehicles/falcon-9/. Accessed 1, August, 2020.

[69] W. D. Graham. Delta III, July 2010. URL https://commons.wikimedia.org/wiki/File:Delta_III.svg. Accessed 1, August, 2020.

Appendices

Appendix A

Detailed Jacobian Matrix Structure

In this Appendix is presented the generic structure of the Jacobian matrix in accordance with

- the flipped Radau pseudospectral method.
- the ordering of the decision variables described in Section 3.3.
- the choice of covector scalars $C_{\lambda,k}^{(p)}$ and $C_{\mu,k}^{(p)}$ presented in Table 3.2.

The generic form of the Jacobian matrix from (3.34) is here reiterated,

$$\mathbf{Jac} = \begin{bmatrix} \nabla \mathcal{J}^N \\ \nabla \boldsymbol{F}^{(1:P)} \\ \nabla \boldsymbol{H}^{(1:P)} \\ \nabla \boldsymbol{\phi}^{(1:P)} \\ \nabla \boldsymbol{\ell}^{(1:P-1)} \end{bmatrix} . \tag{A.1}$$

In this appendix, a Section is dedicated to each group of gradients of the Jacobian matrix in the same order as they appear in (A.1), such that Section A.1 is dedicated to the gradient of the cost functional, $\nabla \mathcal{J}^N$, Section A.2 is dedicated to the gradient of the dynamic defects, $\nabla \boldsymbol{F}^{(1:P)}$, and so on up to Section A.5 which is dedicated to the gradient of the linkage conditions, $\nabla \boldsymbol{\ell}^{(1:P-1)}$.

In order to simplify the syntax, the quantity, $S^{(p)}$ is introduced as

$$S^{(p)} = n_x + N^{(p)}(n_x + n_u) . \tag{A.2}$$

This quantity represents the size of the array containing the decision variables of state and control in a given phase $^{(p)}$.

It is relevant to note that the computation of partial derivatives is taken for granted in this chapter. These derivatives are not trivial, but they can be computed by means of several different methods. By omitting the computation of the derivatives, the length of this chapter is shortened and also no particular differentiation method is highlighted.

A.1 Cost Function Gradient, $\nabla \mathcal{J}^N$

The phase-wise gradient of the cost function, $\nabla_{\boldsymbol{XU}^{(p)}} \mathcal{J}^N$ is

$$\begin{aligned}\nabla_{\boldsymbol{XU}^{(p)}} \mathcal{J}^N = \frac{t_f^{(p)} - t_0^{(p)}}{2} \Bigg[& \frac{2}{t_f^{(p)} - t_0^{(p)}} \boldsymbol{a}^{(p)} \quad w_1^{(p)} \frac{\partial \Psi_1}{\partial \boldsymbol{x}_1^{(p)}} \quad w_1^{(p)} \frac{\partial \Psi_1}{\partial \boldsymbol{u}_1^{(p)}} \quad w_2^{(p)} \frac{\partial \Psi_2}{\partial \boldsymbol{x}_2^{(p)}} \quad w_2^{(p)} \frac{\partial \Psi_2}{\partial \boldsymbol{u}_2^{(p)}} \quad \cdots \\ & w_{N^{(p)}-1}^{(p)} \frac{\partial \Psi_{N^{(p)}-1}}{\partial \boldsymbol{x}_{N^{(p)}-1}^{(p)}} \quad w_{N^{(p)}-1}^{(p)} \frac{\partial \Psi_{N^{(p)}-1}}{\partial \boldsymbol{u}_{N^{(p)}-1}^{(p)}} \quad \frac{2}{t_f^{(p)} - t_0^{(p)}} \boldsymbol{b}^{(p)} + w_{N^{(p)}}^{(p)} \frac{\partial \Psi_{N^{(p)}}}{\partial \boldsymbol{x}_{N^{(p)}}^{(p)}} \quad w_{N^{(p)}}^{(p)} \frac{\partial \Psi_{N^{(p)}}}{\partial \boldsymbol{u}_{N^{(p)}}^{(p)}} \Bigg] \qquad \text{(A.3)} \\ = \nabla \mathcal{J}_{\boldsymbol{XU}}^{(p)} , &\end{aligned}$$

where $\nabla \mathcal{J}_{\boldsymbol{XU}}^{(p)}$ is of size $1 \times S^{(p)}$, and

$$\boldsymbol{a}^{(p)} = \begin{cases} \dfrac{\partial \Phi}{\partial \boldsymbol{x}_0^{(p)}} , & \text{if } p = 1 \\ \boldsymbol{0}_{1 \times n_x} , & \text{if } p \neq 1 \end{cases} , \qquad \boldsymbol{b}^{(p)} = \begin{cases} \boldsymbol{0}_{1 \times n_x} , & \text{if } p \neq P \\ \dfrac{\partial \Phi}{\partial \boldsymbol{x}_{N^{(p)}}^{(p)}} , & \text{if } p = P \end{cases} . \qquad \text{(A.4)}$$

And

$$\frac{\partial \mathcal{J}^N}{\partial t_f^{(p)}} = \begin{cases} \dfrac{1}{2} \displaystyle\sum_k^{N^{(p)}} w_k^{(p)} \Psi_k^{(p)} - \frac{1}{2} \sum_k^{N^{(p+1)}} w_k^{(p+1)} \Psi_k^{(p+1)} , & \text{if } p = 1, 2, \ldots, P-1 \\ \dfrac{1}{2} \displaystyle\sum_k^{N^{(p)}} w_k^{(p)} \Psi_k^{(p)} + \dfrac{\partial \Phi}{\partial t_f^{(p)}} , & \text{if } p = P \end{cases} \qquad \text{(A.5)}$$

$$= \nabla \mathcal{J}_{t_f}^{(p)} . \qquad \text{(A.6)}$$

Ultimately, the vector of concatenated gradients of the cost is

$$\nabla \mathcal{J}^N = \begin{bmatrix} \nabla \mathcal{J}_{\boldsymbol{XU}}^{(1)} & \nabla \mathcal{J}_{\boldsymbol{XU}}^{(2)} & \cdots & \nabla \mathcal{J}_{\boldsymbol{XU}}^{(P)} & \nabla \mathcal{J}_{t_f}^{(1)} & \nabla \mathcal{J}_{t_f}^{(2)} & \cdots & \nabla \mathcal{J}_{t_f}^{(P)} \end{bmatrix} . \qquad \text{(A.7)}$$

Equation (A.7) corresponds to the first row of the matrix illustrated in Figure 3.2.

A.2 Gradient of Dynamic Defects, $\nabla \boldsymbol{F}^{(1:P)}$

The dependency of the dynamic defects with respect to the states and controls can be stated as

$$\nabla_{\boldsymbol{XU}^{(q)}} \boldsymbol{F}^{(p)} = \begin{cases} \nabla \boldsymbol{F}_{\boldsymbol{XU}}^{(p)} , & \text{if } q = p \\ \boldsymbol{0}_{n_x N^{(p)} \times S^{(q)}} , & \text{if } q \neq p \end{cases} , \qquad \text{(A.8)}$$

which is saying that states and controls of a given phase do not influence the dynamics of a different phase. This is favourable because it implies that the Jacobian will be sparse. The dynamic defects for a given phase can be expressed as

$$\nabla \boldsymbol{F}_{\boldsymbol{XU}}^{(p)} = \begin{bmatrix} -\boldsymbol{D}_{10}^{(p)} \boldsymbol{I}_{n_x} & \frac{t_f^{(p)} - t_0^{(p)}}{2} \frac{\partial \boldsymbol{f}_1^{(p)}}{\partial \boldsymbol{x}_1^{(p)}} - \boldsymbol{D}_{11}^{(p)} \boldsymbol{I}_{n_x} & \frac{t_f^{(p)} - t_0^{(p)}}{2} \frac{\partial \boldsymbol{f}_1^{(p)}}{\partial \boldsymbol{u}_1^{(p)}} & \cdots & -\boldsymbol{D}_{1N^{(p)}}^{(p)} \boldsymbol{I}_{n_x} & \boldsymbol{0}_{n_x \times n_u} \\ -\boldsymbol{D}_{20}^{(p)} \boldsymbol{I}_{n_x} & -\boldsymbol{D}_{21}^{(p)} \boldsymbol{I}_{n_x} & \boldsymbol{0}_{n_x \times n_u} & \cdots & -\boldsymbol{D}_{2N^{(p)}}^{(p)} \boldsymbol{I}_{n_x} & \boldsymbol{0}_{n_x \times n_u} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -\boldsymbol{D}_{N^{(p)}0}^{(p)} \boldsymbol{I}_{n_x} & -\boldsymbol{D}_{N^{(p)}1}^{(p)} \boldsymbol{I}_{n_x} & \boldsymbol{0}_{n_x \times n_u} & \cdots & \frac{t_f^{(p)} - t_0^{(p)}}{2} \frac{\partial \boldsymbol{f}_{N^{(p)}}^{(p)}}{\partial \boldsymbol{x}_{N^{(p)}}^{(p)}} - \boldsymbol{D}_{N^{(p)}N^{(p)}}^{(p)} \boldsymbol{I}_{n_x} & \frac{t_f^{(p)} - t_0^{(p)}}{2} \frac{\partial \boldsymbol{f}_{N^{(p)}}^{(p)}}{\partial \boldsymbol{u}_{N^{(p)}}^{(p)}} \end{bmatrix} . \qquad \text{(A.9)}$$

Further, the dependency of the dynamic defects with respect to the initial and final times is

$$\nabla_{t_f^{(q)}} \boldsymbol{F}^{(p)} = \left[\frac{\partial \boldsymbol{\xi}_1^{(p)\,\mathsf{T}}}{\partial t_f^{(q)}} \quad \frac{\partial \boldsymbol{\xi}_2^{(p)\,\mathsf{T}}}{\partial t_f^{(q)}} \quad \cdots \quad \frac{\partial \boldsymbol{\xi}_{N^{(p)}}^{(p)\,\mathsf{T}}}{\partial t_f^{(q)}} \right]^{\mathsf{T}} , \tag{A.10}$$

with

$$\frac{\partial \boldsymbol{\xi}_k^{(p)}}{\partial t_f^{(q)}} = \begin{cases} \frac{1}{2}\boldsymbol{f}_k \,, & \text{if } q = p \\ -\frac{1}{2}\boldsymbol{f}_k \,, & \text{if } q = p-1 \\ \boldsymbol{0}_{n_x \times 1} \,, & \text{if } q \neq p, p-1 \end{cases} . \tag{A.11}$$

Letting the initial and final time gradients of a given phase be

$$\nabla \boldsymbol{F}_{t_0}^{(p)} = \nabla_{t_f^{(p-1)}} \boldsymbol{F}^{(p)} \,, \tag{A.12}$$

$$\nabla \boldsymbol{F}_{t_f}^{(p)} = \nabla_{t_f^{(p)}} \boldsymbol{F}^{(p)} \,. \tag{A.13}$$

Accounting for all phases, the gradient of the dynamic defects can be written as

$$\nabla \boldsymbol{F}^{(1:P)} = \begin{bmatrix} \nabla \boldsymbol{F}_{\boldsymbol{XU}}^{(1)} & \boldsymbol{0}_{n_x N^{(1)} \times S^{(2)}} & \cdots & \boldsymbol{0}_{n_x N^{(1)} \times S^{(P)}} & \nabla \boldsymbol{F}_{t_f}^{(1)} & \boldsymbol{0}_{n_x N^{(1)} \times 1} & \cdots & \boldsymbol{0}_{n_x N^{(1)} \times 1} & \boldsymbol{0}_{n_x N^{(1)} \times 1} \\ \boldsymbol{0}_{n_x N^{(2)} \times S^{(1)}} & \nabla \boldsymbol{F}_{\boldsymbol{XU}}^{(2)} & \cdots & \boldsymbol{0}_{n_x N^{(2)} \times S^{(P)}} & \nabla \boldsymbol{F}_{t_0}^{(2)} & \nabla \boldsymbol{F}_{t_f}^{(2)} & \cdots & \boldsymbol{0}_{n_x N^{(2)} \times 1} & \boldsymbol{0}_{n_x N^{(2)} \times 1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0}_{n_x N^{(P)} \times S^{(1)}} & \boldsymbol{0}_{n_x N^{(P)} \times S^{(2)}} & \cdots & \nabla \boldsymbol{F}_{\boldsymbol{XU}}^{(P)} & \boldsymbol{0}_{n_x N^{(P)} \times 1} & \boldsymbol{0}_{n_x N^{(P)} \times 1} & \cdots & \nabla \boldsymbol{F}_{t_0}^{(P)} & \nabla \boldsymbol{F}_{t_f}^{(P)} \end{bmatrix} . \tag{A.14}$$

In this matrix the sparsity pattern of the dynamic defects is evident. This pattern is represented by blocks of red dots and blue diagonal dots in Fig. 3.2.

A.3 Gradient of Path Constraints, $\nabla \boldsymbol{H}^{(1:P)}$

Similarly to the case of dynamic defects, the states and controls of a given phase do not influence the path constraints of a different phase. The phase-wise path constraint gradient is thus:

$$\nabla \boldsymbol{H}_{\boldsymbol{XU}}^{(p)} = \begin{bmatrix} \boldsymbol{0}_{n_h^{(p)} \times n_x} & \frac{\partial \boldsymbol{h}_1^{(p)}}{\partial \boldsymbol{x}_1^{(p)}} & \frac{\partial \boldsymbol{h}_1^{(p)}}{\partial \boldsymbol{u}_1^{(p)}} & \boldsymbol{0}_{n_h^{(p)} \times n_x} & \boldsymbol{0}_{n_h^{(p)} \times n_u} & \cdots & \boldsymbol{0}_{n_h^{(p)} \times n_x} & \boldsymbol{0}_{n_h^{(p)} \times n_u} \\ \boldsymbol{0}_{n_h^{(p)} \times n_x} & \boldsymbol{0}_{n_h^{(p)} \times n_x} & \boldsymbol{0}_{n_h^{(p)} \times n_u} & \frac{\partial \boldsymbol{h}_2^{(p)}}{\partial \boldsymbol{x}_2^{(p)}} & \frac{\partial \boldsymbol{h}_2^{(p)}}{\partial \boldsymbol{u}_2^{(p)}} & \cdots & \boldsymbol{0}_{n_h^{(p)} \times n_x} & \boldsymbol{0}_{n_h^{(p)} \times n_u} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0}_{n_h^{(p)} \times n_x} & \boldsymbol{0}_{n_h^{(p)} \times n_x} & \boldsymbol{0}_{n_h^{(p)} \times n_u} & \boldsymbol{0}_{n_h^{(p)} \times n_x} & \boldsymbol{0}_{n_h^{(p)} \times n_u} & \cdots & \frac{\partial \boldsymbol{h}_{N^{(p)}}^{(p)}}{\partial \boldsymbol{x}_{N^{(p)}}^{(p)}} & \frac{\partial \boldsymbol{h}_{N^{(p)}}^{(p)}}{\partial \boldsymbol{u}_{N^{(p)}}^{(p)}} \end{bmatrix} , \tag{A.15}$$

and the concatenated matrix for all phases is

$$\nabla \boldsymbol{H}^{(1:P)} = \begin{bmatrix} \nabla \boldsymbol{H}_{\boldsymbol{XU}}^{(1)} & \boldsymbol{0}_{n_h^{(1)} N^{(1)} \times S^{(2)}} & \cdots & \boldsymbol{0}_{n_h^{(1)} N^{(1)} \times S^{(P)}} & \boldsymbol{0}_{n_h^{(1)} N^{(1)} \times P} \\ \boldsymbol{0}_{n_h^{(2)} N^{(2)} \times S^{(1)}} & \nabla \boldsymbol{H}_{\boldsymbol{XU}}^{(2)} & \cdots & \boldsymbol{0}_{n_h^{(2)} N^{(2)} \times S^{(P)}} & \boldsymbol{0}_{n_h^{(2)} N^{(2)} \times P} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0}_{n_h^{(P)} N^{(P)} \times S^{(1)}} & \boldsymbol{0}_{n_h^{(P)} N^{(P)} \times S^{(2)}} & \cdots & \nabla \boldsymbol{H}_{\boldsymbol{XU}}^{(P)} & \boldsymbol{0}_{n_h^{(P)} N^{(P)} \times P} \end{bmatrix} . \tag{A.16}$$

A.4 Event Constraints Gradient, $\nabla \boldsymbol{\phi}^{(1:P)}$

With regards to the event constraints, only the initial and final states of each phase represent dependencies, thus, only two blocks of size $n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}$ appear on the Jacobian of the phase-wise event constraints: one on the left for the dependency on the initial state, and one on the right for the dependency on the final time. In between these two blocks there are only zeros.

$$\nabla \boldsymbol{\phi}_{\boldsymbol{XU}}^{(p)} = \begin{bmatrix} \frac{\partial \boldsymbol{\phi}^{(p)}}{\partial \boldsymbol{x}_0^{(p)}} & \mathbf{0}_{n_{\boldsymbol{\phi}}^{(p)} \times n_{\boldsymbol{x}}} & \mathbf{0}_{n_{\boldsymbol{\phi}}^{(p)} \times n_{\boldsymbol{u}}} & \cdots & \frac{\partial \boldsymbol{\phi}^{(p)}}{\partial \boldsymbol{x}_{N^{(p)}}^{(p)}} & \mathbf{0}_{n_{\boldsymbol{\phi}}^{(p)} \times n_{\boldsymbol{u}}} \end{bmatrix}. \tag{A.17}$$

In order to account for all phases, a diagonal concatenation is in place, as

$$\nabla \boldsymbol{\phi}^{(1:P)} = \begin{bmatrix} \nabla \boldsymbol{\phi}_{\boldsymbol{XU}}^{(1)} & \mathbf{0}_{n_{\boldsymbol{\phi}}^{(1)} \times S^{(2)}} & \cdots & \mathbf{0}_{n_{\boldsymbol{\phi}}^{(1)} \times S^{(P)}} & \mathbf{0}_{n_{\boldsymbol{\phi}}^{(1)} \times P} \\ \mathbf{0}_{n_{\boldsymbol{\phi}}^{(2)} \times S^{(1)}} & \nabla \boldsymbol{\phi}_{\boldsymbol{XU}}^{(2)} & \cdots & \mathbf{0}_{n_{\boldsymbol{\phi}}^{(2)} \times S^{(P)}} & \mathbf{0}_{n_{\boldsymbol{\phi}}^{(2)} \times P} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}_{n_{\boldsymbol{\phi}}^{(P)} \times S^{(1)}} & \mathbf{0}_{n_{\boldsymbol{\phi}}^{(P)} \times S^{(2)}} & \cdots & \nabla \boldsymbol{\phi}_{\boldsymbol{XU}}^{(P)} & \mathbf{0}_{n_{\boldsymbol{\phi}}^{(P)} \times P} \end{bmatrix}. \tag{A.18}$$

A.5 Gradient of the Linkage Conditions, $\nabla \boldsymbol{\ell}^{(1:P-1)}$

Finally, for the linkage conditions, the matrices for the left and right pairs of phases are defined as

$$\nabla \Delta \boldsymbol{x}_l^{(p)} = \begin{bmatrix} \mathbf{0}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}} & \mathbf{0}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}} & \mathbf{0}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{u}}} & \cdots & -\boldsymbol{I}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}} & \mathbf{0}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{u}}} \end{bmatrix} \tag{A.19}$$

$$\nabla \Delta \boldsymbol{x}_r^{(p)} = \begin{bmatrix} \boldsymbol{I}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}} & \mathbf{0}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}} & \mathbf{0}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{u}}} & \cdots & \mathbf{0}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}} & \mathbf{0}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{u}}} \end{bmatrix} \tag{A.20}$$

where $\boldsymbol{I}_{n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}}$ is the identity matrix of size $n_{\boldsymbol{x}} \times n_{\boldsymbol{x}}$. Invariably, the full gradient matrix for the linkage conditions is a diagonal concatenation of of the pairs of matrices expressed above.

$$\nabla \boldsymbol{\ell}^{(1:P-1)} = \begin{bmatrix} \nabla \Delta \boldsymbol{x}_l^{(1)} & \nabla \Delta \boldsymbol{x}_r^{(1)} & \mathbf{0}_{n_{\boldsymbol{x}} \times S^{(3)}} & \cdots & \mathbf{0}_{n_{\boldsymbol{x}} \times S^{(P-1)}} & \mathbf{0}_{n_{\boldsymbol{x}} \times S^{(P)}} & \mathbf{0}_{n_{\boldsymbol{x}} \times P} \\ \mathbf{0}_{n_{\boldsymbol{x}} \times S^{(1)}} & \nabla \Delta \boldsymbol{x}_l^{(2)} & \nabla \Delta \boldsymbol{x}_r^{(2)} & \cdots & \mathbf{0}_{n_{\boldsymbol{x}} \times S^{(P-1)}} & \mathbf{0}_{n_{\boldsymbol{x}} \times S^{(P)}} & \mathbf{0}_{n_{\boldsymbol{x}} \times P} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0}_{n_{\boldsymbol{x}} \times S^{(1)}} & \mathbf{0}_{n_{\boldsymbol{x}} \times S^{(2)}} & \mathbf{0}_{n_{\boldsymbol{x}} \times S^{(3)}} & \cdots & \nabla \Delta \boldsymbol{x}_l^{(P-1)} & \nabla \Delta \boldsymbol{x}_r^{(P-1)} & \mathbf{0}_{n_{\boldsymbol{x}} \times P} \end{bmatrix} \tag{A.21}$$

It can be noted that the first phase is never the "right" pair of a link and that the last phase is never the "left" pair of any link. This is because the algorithm invariably assumes that the phases are sequential.