

Copyright

by

Etienne Pellegrini

2017

The Dissertation Committee for Etienne Pellegrini
certifies that this is the approved version of the following dissertation:

**Multiple-Shooting Differential Dynamic Programming
with Applications to Spacecraft Trajectory
Optimization**

Committee:

Ryan P. Russell, Supervisor

Maruthi R. Akella

Efstathios Bakolas

Max Cerf

Brandon A. Jones

**Multiple-Shooting Differential Dynamic Programming
with Applications to Spacecraft Trajectory
Optimization**

by

Etienne Pellegrini

DISSERTATION

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

The University of Texas at Austin
August 2017

To optimally go where no man has gone before.

Multiple-Shooting Differential Dynamic Programming with Applications to Spacecraft Trajectory Optimization

by

Etienne Pellegrini, Ph.D.

The University of Texas at Austin, 2017

SUPERVISOR: Ryan P. Russell

The optimization of spacecraft trajectories has been, and continues to be, critical for the development of modern space missions. Longer flight times, continuous low-thrust propulsion, and multiple flybys are just a few of the modern features resulting in increasingly complex optimal control problems for trajectory designers to solve. In order to efficiently tackle such challenging problems, a variety of methods and algorithms have been developed over the last decades. The work presented in this dissertation aims at improving the solutions and the robustness of the optimal control algorithms, in addition to reducing their computational load and the amount of necessary human involvement. Several areas of improvement are examined in the dissertation. First, the general formulation of a Differential Dynamic Programming (DDP) algorithm is examined, and new theoretical developments are made in order to achieve a multiple-shooting formulation of the method. Multiple-shooting transcriptions have been demonstrated to be beneficial to both direct

and indirect optimal control methods, as they help decrease the large sensitivities present in highly nonlinear problems (thus improving the algorithms' robustness), and increase the potential for a parallel implementation. The new Multiple-Shooting Differential Dynamic Programming algorithm (MDDP) is the first application of the well-known multiple-shooting principles to DDP. The algorithm uses a null-space trust-region method for the optimization of quadratic subproblems subject to simple bounds, which permits to control the quality of the quadratic approximations of the objective function. Equality and inequality path and terminal constraints are treated with a general Augmented Lagrangian approach. The choice of a direct transcription and of an Augmented Lagrangian merit function, associated with automated partial computations, make the MDDP implementation flexible, requiring minimal user effort for changes in the dynamics, cost and constraint functions. The algorithm is implemented in a general, modular optimal control software, and the performance of the multiple-shooting formulation is analyzed. The use of quasi-Newton approximations in the context of DDP is examined, and numerically demonstrated to improve computational efficiency while retaining attractive convergence properties.

The computational performance of an optimal control algorithm is closely related to that of the integrator chosen for the propagation of the equation of motion. In an effort to improve the efficiency of the MDDP algorithm, a new numerical propagation method is developed for the Kepler, Stark, and three-body problems, three of the most commonly used dynamical models for spacecraft trajectory optimization. The method uses a time reg-

ularization technique, the generalized Sundman transformation, and Taylor Series developments of equivalents to the f and g functions for each problem. The performance of the new method is examined, and specific domains where the series solution outperforms existing propagation methods are identified.

Finally, because the robustness and computational efficiency of the MDDP algorithm depend on the quality of the first- and second-order State Transition Matrices, the three most common techniques for their computation are analyzed, in particular for low-fidelity propagation. The propagation of variational equations is compared to the complex step derivative approximation and finite differences methods, for a variety of problems and integration techniques. The subtle differences between variable- and fixed-step integration for partial computation are revealed, common pitfalls are observed, and recommendations are made for the practitioner to enhance the quality of state transition matrices.

Table of Contents

Abstract	v
List of Tables	xiii
List of Figures	xiv
Chapter 1. Introduction	1
1.1 Problem Definition and Motivation	1
1.2 Literature Review	3
1.2.1 Optimal Control for Trajectory Optimization	3
1.2.1.1 Indirect and Direct Optimization	4
1.2.1.2 Transcription Methods	7
1.2.1.3 Nonlinear Programming	11
1.2.1.4 Differential Dynamic Programming	23
1.2.1.5 Motivations for the Multiple-Shooting Differential Dynamic Programming Algorithm	27
1.2.2 Numerical Partial Derivatives	28
1.2.3 Numerical Propagation of Spacecraft Trajectories	32
1.3 Outline of the Dissertation	36
1.4 Summary of Contributions	38
Chapter 2. Mathematical Background	41
2.1 Notations and Conventions	41
2.1.1 State-Transition Matrices	43
2.2 Dynamical Models	44
2.2.1 Two-Body Problem	44
2.2.1.1 Stark Problem	45
2.2.2 Circular Restricted Three-Body Problem (CRTBP)	47
2.2.3 The Generalized Sundman Transformation	50
2.2.3.1 Presentation and Historical Survey	50

2.2.3.2	Application of the Sundman Transformation to the Kepler and Stark Problems	52
2.2.3.3	Application to the CRTBP	57
Chapter 3.	Multiple-Shooting Differential Dynamic Program- ming	59
3.1	The Multiple-Phase Optimal Control Problem Formulation . .	60
3.2	Augmented Lagrangian Algorithm for Constrained Optimization	67
3.2.1	Equality Constrained Problems	68
3.2.2	Extension to Inequality Constraints	71
3.2.3	Application to the MDDP algorithm	75
3.3	Solving the Multiple-Shooting Problem	77
3.3.1	Single-leg Quadratic Expansions	78
3.3.2	Multi-leg Quadratic Expansions	84
3.3.3	Lagrange Multipliers Update	87
3.4	Overview of the MDDP algorithm	90
3.4.1	Trust-Region Algorithm	95
3.4.2	Cost Ratio Computation and Trust-Region Update . .	97
3.4.3	Convergence Tests	100
3.5	Estimating the Second-Order State Transition Matrices . . .	103
3.5.1	Quasi-Newton Methods	104
3.5.2	Choosing an Adapted Quasi-Newton Update: the Sym- metric Rank 1 Update	107
Chapter 4.	Applications of the Multiple-Shooting Differential Dynamic Algorithm	110
4.1	MDDP Algorithm Testing Framework	111
4.1.1	A Collection of Optimal Control Problems	111
4.1.1.1	Quadratic-Linear Problem	111
4.1.1.2	Brachistochrone problem	112
4.1.1.3	Van Der Pol Oscillator	112
4.1.1.4	One-Dimensional Landing Problem	113
4.1.1.5	Two-Dimensional Orbit Transfer	114
4.1.1.6	Three-Dimensional Two-Body Problem	115
4.1.2	Settings & Parameters	116

4.2	Validation of the MDDP algorithm	116
4.2.1	Validation of the Quadratic Expansions and Updates for the Single-Leg Solver	117
4.2.2	Validation of the Treatment of Terminal Constraints . .	118
4.2.3	Validation of the Treatment of Path Constraints . . .	121
4.3	Performance Analysis of the Multiple-Shooting Formulation . .	123
4.3.1	Multiple-Revolution Orbit transfer	129
4.3.2	Quasi-Newton Numerical Results	130
Chapter 5.	On the Computation and Accuracy of Trajectory State Transition Matrices	137
5.1	Methods	138
5.1.1	Variational Equations	138
5.1.2	Complex Step Derivative	141
5.1.3	Finite Differences	145
5.1.3.1	Generic Finite Differences	145
5.1.3.2	Second-Order Finite Differences	148
5.2	Applications	149
5.2.1	The Testing Framework	150
5.2.1.1	Test Cases	150
5.2.1.2	Fixed Path Feature	153
5.2.1.3	The Linesearch Application	154
5.2.1.4	Tuning the Perturbation Magnitude	156
5.2.2	The Caveats of Variable-Step Integration	157
5.2.2.1	Path with Variational Equations	158
5.2.2.2	Partials of the Variable-Step Integration	161
5.2.2.3	Discontinuity of the Integration Path	164
5.2.3	Accuracy and Timings of the Partial Computations . .	167
5.3	Conclusions	172
Chapter 6.	F and G series solutions to the Kepler, Stark, and Circular Restricted Three-Body Problems	175
6.1	Introduction	175
6.2	Development of the Series Formulations	177
6.2.1	<i>F</i> and <i>F</i> Stark Series	178

6.2.2	<i>F</i> and <i>G</i> CRTBP Series	184
6.3	Numerical Validation and Comparison to Other Integrators . .	188
6.3.1	<i>F</i> and <i>G</i> Stark Series	188
6.3.1.1	The Modern Taylor Series Method	189
6.3.1.2	Accuracy Estimation	190
6.3.1.3	Study of the Generalized Sundman Transformation for the Unperturbed 2-Body Problem . .	193
6.3.1.4	Runtime Comparisons Between the Taylor Series and the RKF8 Integrator	197
6.3.1.5	Application: Low-Thrust Orbital Transfer . . .	202
6.3.2	<i>F</i> and <i>G</i> CRTBP Series	205
6.3.2.1	Framework	206
6.3.2.2	Variable-Step Integration	208
6.3.2.3	Fixed-Step Integration	213
6.4	Conclusions	216
Chapter 7. Conclusions		219
Appendices		225
Appendix A. Orbital Period in τ for Different Sundman Transformations		226
A.1	Case $d\tau = cdt$	227
A.2	Case $d\tau = crdt$	227
A.3	Case $d\tau = cr^2dt$	228
A.4	Case $d\tau = cr^{3/2}dt$	229
Appendix B. Algorithm Used for the Performance Comparison of all Integrators Applied to an Example Low-Thrust Orbital Transfer		230
Appendix C. Implementing the <i>F</i> and <i>G</i> Stark Series with a Symbolic Manipulator		232
Appendix D. List of Publications		235
D.1	Refereed Journal Publications	235
D.2	Conference Proceedings	235

Bibliography **237**

Vita **303**

List of Tables

2.1	τ -period for different values of α	55
4.1	Tuning parameters and their default values	117
4.2	Number of iterations for the collection of problems	128
5.1	Initial conditions for the 3-body periodic orbits	151
5.2	Perturbation magnitude for the finite difference methods	158
6.1	Initial conditions for the four test scenarios	207

List of Figures

2.1	Sketch of the inertial frame CRTBP	46
2.2	Effect of the Sundman power law. $a = 1, e = 0.7$	56
2.3	τ -period vs. orbital elements	56
3.1	Discretization of the optimal control problem for two phases .	66
3.2	Inner loop of the MDDP algorithm	88
3.3	Structure of the MDDP algorithm	89
3.4	Heuristic multiplier $\kappa(\rho)$ for trust-region radius update	99
4.1	Solution of the quadratic linear problem. $J_1 = 1.2807$	118
4.2	Solutions to the constrained and unconstrained Van Der Pol oscillator. $J_1 = 2.8268$ for the unconstrained solution, $J_1 = 2.8658$ for the constrained solution.	119
4.3	Solutions to the 1D landing problem with active and inactive terminal constraint on velocity. $J_1 = -m(t_f) = -1$ MU for the inactive solution, $J_1 = -m(t_f) = -0.39527$ MU for the active solution.	120
4.4	Solutions to the 3D Hohmann transfer with active and inactive terminal constraint on circularity. $J_1 = -m(t_f) = -0.95581$ when the circular constraint is inactive, $J_1 = -m(t_f) = -0.92401$ when it is active.	121
4.5	Solutions to the constrained and unconstrained Brachistochrone problem. $J_1 = t_f = 0.82446$ TU for the unconstrained solution, $J_1 = t_f = 0.91495$ TU for the constrained solution 1, and $J_1 = t_f = 0.85870$ TU for the constrained solution 2.	122
4.6	Solutions to the constrained and unconstrained Van Der Pol oscillator. $J_1 = 2.8658$ for the unconstrained problem, $J_1 = 4.1716$ for the constrained problem.	123
4.7	Solutions to the 3D Hohmann transfer with inactive and active path constraint. The final cost is $J_1 = -0.92401$ for the unconstrained solution, and $J_1 = -0.90887$ for the constrained problem.	124
4.8	Solutions of the quadratic linear problem with 10 legs. $J = 1.2807124$	

4.9	Solutions of the path constrained and unconstrained minimum-time Van Der Pol oscillator. $J_2 = t_f = 2.6889$ TU for the unconstrained solution, and $J_2 = t_f = 5.8242$ TU for the constrained problem.	126
4.10	Solutions of the minimum-fuel (2DS3 and 2DS4) and minimum-thrust (2DS1 and 2DS2) 2D spacecraft problem. $J_1 = -m(t_f) = -0.90893$ for the minimum-fuel problem, and $J_2 = 2.6449$ for the minimum thrust problem.	127
4.11	Solution of the orbital transfer problem	130
4.12	Sensitivity metric as computed in Eq. (4.27)	130
4.13	Speedup and iteration ratio of the quasi-Newton scenarios for the Brachistochrone problems	131
4.14	Speedup and iteration ratio of the quasi-Newton scenarios for the Van Der Pol problems	132
4.15	Speedup and iteration ratio of the quasi-Newton scenarios for the one-dimensional landing problem	133
4.16	Speedup and iteration ratio of the quasi-Newton scenarios for the two-dimensional spacecraft problems	134
4.17	Speedup and iteration ratio of the quasi-Newton scenarios for the three-dimensional spacecraft problems	135
4.18	Maximum speedup per iteration	135
5.1	Three-body periodic orbits test cases	152
5.2	The two-body orbits	153
5.3	Accuracy results for $\epsilon = 10^{-3}$, when varying the perturbation magnitude used in the finite difference methods	158
5.4	TB2 and its sensitivities integrating the state only path or the state and variational equations path	159
5.5	Results of the linesearch application for TB2, using the RKF(7)8 and the variational equations	160
5.6	Linesearch results for PO3 with $\epsilon = 10^{-3}$	162
5.7	RMS of the linesearch results for PO3	164
5.8	Results of the linesearch using the variational equations and the CSD method for PO3 and its perturbed version, with $\epsilon = 10^{-3}$	166
5.9	Results of the linesearch using the central difference method for PO3 and its perturbed version, with $\epsilon = 10^{-3}$	167
5.10	Linesearch results for fixed and variable-step integration of TB2, $\epsilon = 10^{-6}$	168
5.11	Accuracy	169

5.12	Timings of the different methods for TB2	170
5.13	Accuracy and timings results for PO2	172
6.1	Effect of orbit location on the local truncation error. $a = 1$, $e = 0.6$, $\mathbf{p} = \mathbf{0}$, $\mu = 1$	195
6.2	Effect of α on the local truncation error. α equally spaced using $\Delta\alpha = 0.01$, $a = 1$, $e = 0.6$, $\mathbf{p} = \mathbf{0}$, $\mu = 1$	196
6.3	Timing results when varying integration parameters, for all in- tegrators	199
6.4	Application: low-thrust orbital transfer	204
6.5	Results of the application: Speedup of Taylor series solutions compared to time-velocity RKF8	204
6.6	Results of the application: Speedups and accuracy of Taylor series solutions compared to τ -velocity RKF8	205
6.7	Scenario 1, with different Sundman transformations. $\epsilon_{\text{glob}} \leq 10^{-6}$	208
6.8	Scenario 2, with different Sundman transformations. $\epsilon_{\text{glob}} \leq 10^{-6}$	208
6.9	Scenario 3, with different Sundman transformations. $\epsilon_{\text{glob}} \leq 10^{-6}$	209
6.10	Scenario 4, with different Sundman transformations. $\epsilon_{\text{glob}} \leq 10^{-6}$	210
6.11	Speedups of the variable-step F and G CRTBP propagator com- pared to the RKF(7)8. No Sundman transformation is used ($s = 1$). For Scenario 3, for $\epsilon_{\text{thresh}} = 10^{-12}$, neither the F and G series nor the RKF converged	212
6.12	Speedups of the fixed-step F and G CRTBP propagator com- pared to the RKF8	214
6.13	Number of steps necessary to achieve the prescribed tolerance $\epsilon_{\text{thresh}} = 10^{-6}$ using fixed-step integration.	215

Chapter 1

Introduction

1.1 Problem Definition and Motivation

The exploration of space has benefited from many technological advances in the past decades, allowing humans and their instruments to reach farther, more varied, and ultimately more scientifically valuable destinations. Low-thrust, high-specific-impulse propulsion systems in particular permit the design of efficient exploration missions, consuming less propellant and therefore allowing for larger payloads, shorter flight times, or the visit of previously unreachable objects. Examples of successful use of continuous low-thrust propulsion system include Deep Space 1 [276], Smart 1 [269, 169], Hayabusa [328, 163], and Dawn [278]. The longer flight times and the continuous operation of the propulsion system associated with low-thrust missions significantly increase the complexity of the optimization problem. The approximation of continuous thrust requires a finer discretization, resulting in large-scale non-linear programs, and more complex dynamical models, increasing the computational load of the algorithms. Moreover, modern mission analysts strive to utilize advanced dynamical concepts such as multi-body interactions to further lower the required energy [180, 174]. These models usually amplify the nonlinearity of the trajectories, increasing the sensitivity of the algorithm and their

computational complexity. Fortunately, advances in digital computing, as well as in optimal control techniques, allow practitioners to use a variety of new solvers designed for trajectory optimization. These solvers aim at improving the solutions and the robustness of the algorithms, but also at reducing the computational time and the amount of necessary human effort. A review of modern spacecraft trajectory optimization methods is given in [Section 1.2.1](#).

The work presented in this dissertation aims at improving the state-of-the-art of modern optimal control solvers. Two main axes of improvement are defined: 1) the computational efficiency of the numerical methods, represented by the runtime on a personal workstation, and 2) the robustness of the optimal control algorithm, defined as the reliability of the algorithm under variations of the problem complexity, initial conditions and tuning parameters. Robust optimal control algorithms allow practitioners to use automated initial guess generation methods, and therefore permit a reduction of the amount of necessary human effort. A new algorithm is presented which combines a multiple-shooting transcription of the optimal control problem with Differential Dynamic Programming (DDP) principles, in order to improve the robustness of existing DDP methods, and increase the potential for a parallel implementation. The resulting algorithm, dubbed Multiple-Shooting Differential Dynamic Programming or MDDP algorithm, is the first application of the well-known multiple-shooting principles to the DDP framework. The algorithm uses a null-space trust-region method for the optimization of the controls subject to simple bounds, and an Augmented Lagrangian strategy for quality and inequality path and terminal constraints. MDDP uses a number

of strategies first developed for the Hybrid Differential Dynamic Programming (HDDP) algorithm of Lantoine and Russell [178, 179], in particular the use of first- and second-order State-Transition Matrices (STMs) for sensitivity propagation. Because MDDP is based on the computation of first- and/or second-order partials, the quality of the numerical methods for sensitivity computation is crucial to their robustness and efficiency. Three common techniques available to practitioners are presented and compared, and recommendations are made to improve the quality of the derivatives. New numerical solutions to the Stark and Circular Restricted Three-Body problems are developed, accelerating the numerical propagation of the equations of motion of these two dynamical models, which are among the most used in spacecraft trajectory design.

1.2 Literature Review

1.2.1 Optimal Control for Trajectory Optimization

The optimal control problem consists in finding a set of input functions that minimize a functional performance index subject to differential and algebraic constraints [155]. This area of study was born in 1696 with Bernouilli's publication of the Brachistochrone problem in *Acta Eruditorum*. The study of a multitude of challenging optimal control problems, including that of optimal spacecraft trajectories, has occupied mathematicians ever since [41, 257, 149]. Optimal control solution methods are usually classified into one of two categories: 1) indirect methods, developed first historically, use the calculus of variations to solve for the necessary conditions for optimality and transform the

optimal control problem in a boundary value problem [183, 56, 155]; 2) direct methods discretize and parameterize the state and controls in order to transcribe [154] the optimal control problem into a nonlinear parameter optimization problem, which aims at directly minimizing the performance index, while satisfying the constraints. Numerical methods for both direct and indirect approaches can use a variety of Non Linear Programming (NLP) techniques to solve for the state, controls, and sometimes co-states [127, 109, 27, 47, 230]. In addition, DDP techniques, based on Bellman’s Principle of Optimality [11, 12], are hybrid methods which discretize the problem to solve for the states and controls directly, but exploit the time dependent dynamical structure to solve a sequence of quadratic approximations, and provide optimal feedback laws for the controls. A brief review of state-of-the-art trajectory optimization methods, as well as examples of solvers and their use, are given in this section. The motivations for a multiple-shooting DDP algorithm for the optimization of low-thrust spacecraft trajectories are explained.

1.2.1.1 Indirect and Direct Optimization

Trajectory optimization algorithms are typically divided into two classes, which differ by their approach of the problem. Indirect methods use the calculus of variations, and solve the problem by introducing co-states, and deriving co-state equations, controls equations, and transversality conditions [149]. The optimal control problem is reduced to a boundary value problem where the unknowns are the states and co-states at a single instant in time (usually the initial or final time), subject to differential constraints (the dynamics and

co-states equations) and boundary conditions (the transversality conditions). The controls are computed explicitly or implicitly from the controls equation, according to what is known in the spacecraft trajectory optimization community as primer vector theory [182, 294], and are therefore removed from the unknown vector [25]. When the boundary value problem is solved, a continuous control history is computed from the controls equation, and indirect methods are therefore more accurate than direct methods, which parameterize and approximate the controls [272]. Moreover, in an indirect approach to the optimal control problem, a single value of the co-states vector replaces the control history as decision variable, drastically reducing the dimension of the problem. With the exception of simple problems of little practical use, analytical solutions of boundary value problems can prove extremely hard or impossible to obtain. Numerical methods have to be devised, such as gradient methods [165, 55, 208], or integral-equation methods [164]. Most modern indirect solvers now employ transcription methods such as the ones described in Section 1.2.1.2 in order to parameterize the boundary value problem, and use NLP techniques to solve it [32, 335].

One of the main drawbacks of indirect methods is their reduced radius of convergence, when compared to their direct counterparts. The coupling of the dynamics and co-states equations increases the possibility of divergence of the trajectory when poor guesses are made for the initial conditions, and the co-state equations are typically very sensitive to initial conditions [56]. Techniques used to alleviate this requirement for a good initial guess include homotopy methods, which solve a sequence of problems of increasing diffi-

culty, using the solution of the simpler problem as an initial guess for the next optimization [58, 21]. Some direct solvers can also approximate the co-states, and the solution computed by such methods can be used as an initial guess to the indirect method [99, 178]. Morevoer, many indirect methods only satisfy the necessary conditions for optimality. Checking for the more complicated sufficient conditions necessitates additional effort, whereas direct methods typically have this information readily available. Finally, the indirect approach requires the derivation of the co-state equations and transversality conditions for each desired problem, which reduces its flexibility [27]. The use of automatic differentiation software can partly alleviate this requirement [206].

Direct optimal control methods do not use the calculus of variations, but instead discretize the optimal control problem and parameterize the state and the discrete controls. The parameterization can be done using piecewise constant or more complicated polynomial forms, the coefficients of which become the unknowns of a nonlinear program. NLP techniques are used in order to directly drive the objective function towards a local extremum, while satisfying the differential and algebraic constraints [298, 170]. Direct methods usually generate a larger number of decision variables than their indirect counterparts, since the controls at each segment are parameterized, whereas indirect methods only solve for the co-states at a single instant in time [154]. This increased complexity explains that the development of direct optimal control solvers follows the evolution of digital computing and the growth of NLP methods. Generally, computational complexity of the transcribed NLP

rises quadratically or cubically with the number of unknowns [217], which in a direct method is proportional to the number of discretization steps. Moreover, the parameterization of the controls leads to approximations of the control history, and therefore the solutions computed via direct methods are less accurate than their indirect counterparts [270, 335]. The popularity of direct methods is largely due to their greater radius of convergence, making them more robust to initial guesses, an undeniable advantage for the optimization of sensitive systems such as interplanetary trajectories. Direct methods are also more versatile and often easier to implement since they do not require the derivation of the co-state equations and transversality conditions for every new problem [25].

1.2.1.2 Transcription Methods

Most numerical methods for the solution of optimal control problems require the transcription of the time-continuous problem into a nonlinear program. The most common approaches used with both direct and indirect methods are single-shooting, multiple-shooting, and collocation [272].

Indirect single-shooting methods solve the boundary value problem by making an initial guess of the unknown parameters (states and co-states), and numerically integrating the dynamics and co-states equations to compute a final constraint violation vector. NLP algorithms can be used to find the root of the constraint violation vector, thus satisfying the transversality conditions, and yielding an optimal control policy [164, 32]. When the controls can be obtained explicitly from the primer vector theory, classic initial value methods

can be used for the integration, while a differential-algebraic solver is used for implicit definition of the control history [25]. Indirect single-shooting methods present the typical disadvantages of indirect methods, such as high sensitivities and a small radius of convergence, and necessitate excellent initial guesses in order to converge [281, 238, 56, 90].

Direct single-shooting methods discretize the trajectory in order to parameterize the control history, and make an initial guess for the parameters before numerically integrating the resulting trajectory. The parameters are then changed so as to directly drive the cost down, while satisfying the constraints [56, 127]. The “Program to Optimize Simulated Trajectories” (POST) is an example of software implementing a direct single-shooting method [48]. Direct single-shooting also is subject to sensitivity issues, when a small change in the parameters, especially those corresponding to the start of the trajectory, results in a highly nonlinear change at its end.

The main idea behind multiple-shooting is to break up the time interval in a succession of smaller intervals, and to apply single-shooting methods to each of the subintervals. The states and/or co-states at the initial time of each subinterval become optimization parameters, and continuity of the solution is ensured by adding linkage constraints [164, 238]. By reducing the time interval over which the single-shooting method is applied, sensitivity and robustness to the initial guesses are also reduced [42, 58]. Moreover, the structure of multiple-shooting methods allows for a parallel implementation, as the solution can be integrated over each subinterval independently [29]. An obvious drawback of the multiple-shooting approach is an increase in the dimension of

the problem to be solved by the NLP solver, since the initial states and/or co-states of each subinterval become decision variables, and continuity constraints are added. Indirect multiple-shooting methods were introduced first, to increase the robustness of the algorithms to initial guesses [43, 91, 92, 253, 315].

The BNDSCO software successfully implements indirect multiple-shooting [234]. Bock and Plitt [42] successfully applied the multiple-shooting principles to a direct method, and numerous multiple-shooting methods, both direct and indirect, have been developed since [232, 18, 66, 63, 171, 29, 94]. However, multiple-shooting principles have yet to appear in the differential dynamic programming literature. As is shown in Chapter 3, the application of the multiple-shooting transcription to DDP is more complex than for classic direct or indirect approaches, as new feedback equations must be derived in order to appropriately satisfy all linkage and boundary conditions.

In contrast with direct- and multiple-shooting methods, collocation techniques do not require explicit integration of the state and co-state equations [25]. Instead, in traditional collocation methods, the time interval is discretized at the collocation points, and piecewise polynomials are used to parameterize the unknowns at every segment (states and controls for direct collocation, states and co-states for indirect collocation). A number of different choices for the polynomials have been studied, corresponding to different implicit integration schemes. For instance, piecewise Chebyshev polynomials are used in the software CHEBYTOP [158, 143], Hermite polynomials are used in Hermite-Simpson schemes [146], and higher order methods such as Gauss-Lobatto quadrature rules [148] provide more robust and efficient ap-

proximations. In collocation methods, the dynamical equations (state or state and co-state) are enforced through the creation of additional constraints at the collocation points, and the resulting root-finding problem can be solved using classical nonlinear programming [30]. Collocation methods were first created for solving two-point boundary value problems [293], similar to those encountered when employing an indirect approach to the optimal control problem [93]. However, as for other transcription methods, direct methods have been the focus of many of the modern applications of collocation techniques [158, 170, 146, 97, 334, 24, 148]. Examples of trajectory optimization software using traditional direct collocation methods include ALTOS [286], OTIS [333, 280], and the powerful $\text{S}\text{O}\text{C}\text{S}$ [31], developed by the Boeing Company, which has been used for a number of trajectory optimizations, such as lunar fly-bys [28], very low-thrust optimization [26], or interplanetary trajectories [24].

More recently, a new type of collocation method has emerged in the optimal control community: pseudospectral methods, also called orthogonal collocation methods, use global orthogonal polynomials instead of piecewise polynomials, allowing for a better convergence of the approximation as a function of the number of collocation points [272]. These methods were first introduced for the solution of computational fluid dynamics problems [65], and a number of specialized pseudospectral methods for trajectory optimization have since been developed, for direct or indirect approaches [33, 98, 100, 290, 96, 122]. The methods once again differ by the chosen type of polynomials and their associated collocation points. Another attractive property of certain direct

pseudospectral methods is the possibility of estimating the co-states of the dual problem [99, 17]. Trajectory optimization software implementing pseudospectral methods include DIDO [289] and GPOPS – III [245, 138], and a survey of these methods can be found in Ref. [291].

More information concerning the classification of methods and transcriptions can be found in survey papers such as Refs. [335, 25, 79], or Refs. [272, 325], which focus on direct collocation methods.

1.2.1.3 Nonlinear Programming

To tackle challenging optimization problems such as those resulting from a transcribed optimal control problem, a variety of nonlinear programming (NLP) solvers have been developed over the last several decades [27]. Nonlinear programming, also called numerical optimization, parameter optimization, or nonlinear optimization, has a vast history, and this section provides a brief summary of the NLP methods used for trajectory optimization. Only local methods of optimization are considered; global trajectory optimization is a different field entirely, and its state-of-the-art is not included here. More information about the theory of nonlinear programming, such as necessary and sufficient conditions for optimality, as well as details on practical methods for optimization can be found in textbooks such as Refs. [127, 109, 27, 47, 230].

Nonlinear programs choose a finite number of variables such that a scalar objective function is minimized, while satisfying algebraic constraints [155]. Optimization and root-finding problems are closely related [127], and

most optimization methods are based on the computation of first- and sometimes second-order sensitivities of the objective function with respect to the input variables (e.g. Newton’s methods). These sensitivities can be tedious to obtain and are generally expensive to compute. [Section 1.2.2](#) introduces different numerical methods for the obtention of first- and second-order State Transition Matrices (STMs), essential to the application of the MDDP algorithm. [Chapter 5](#) is dedicated to the evaluation of the performance of said methods, in terms of accuracy and computational speed.

Most algorithms for the solution of n -dimensional unconstrained nonlinear optimization problems, designated as “step-length-based” methods, rely on two main steps: 1) the computation of a search direction, and 2) the computation of the length of the step taken along the search direction [127].

Line-search algorithms, or one-dimensional optimization, are used in nonlinear programming in order to ensure sufficient decrease in the objective function along the search direction [237]. A number of different algorithms for univariate minimization are used, such as dichotomy, Fibonacci and Golden ratio searches [167], or polynomial interpolation [83, 85]. Different stopping conditions can be used, such as exact line-searches [82], the Goldstein-Armijo principles [135], or the Wolf-Powell conditions [348, 262]. A line-search requires simple function evaluations, and the quality of a line-search is often dependent on the chosen initial step-length [127].

A typical method for the computation of a search direction consists in approximating the nonlinear objective function using its Taylor series, and

computing a descent direction for the approximate model. The simplest model for the objective function is a linear approximation, which requires first-order information only. Steepest-descent methods, which date back to Cauchy [69], pick the search direction to be along the negative gradient. They have been demonstrated to be excruciatingly slow for some problems [237]. Conjugate-gradient methods, developed first for linear systems by Hestenes and Stiefel [151], and extended to nonlinear programs with updates such as the Fletcher-Reeves [105] or Polak-Ribiere [254] formulas, use only gradient information, but dramatically increase the rate of convergence compared to the steepest-descent method [194].

When first- and second-order information are available, Newton's method can be used to produce quadratically convergent algorithms, given that the initial solution is close enough to the extremum [237]. Approximating the nonlinear function with a convex quadratic form is justified when close to the minimum: the Hessian of the nonlinear objective function is positive-definite at its local minima. For quadratic objective functions, Newton's method converges in one iteration, without requiring a line-search. However, for nonlinear programming, a line-search is needed to ensure sufficient decrease of the objective function. Moreover, when away from a local minimum, there is no guarantee for the Hessian to be convex. Therefore, a number of modified Newton methods have been devised to ensure descent directions, for instance based on eigenvalue or Cholesky decompositions [139, 124].

Quasi-Newton Methods The computational cost of the second-order information can sometimes be prohibitive: for instance, when the number of decision variables is large, or when exact derivatives are unavailable and approximate techniques are required along with expensive function evaluations. The evaluation of the second-order partials often represents the most computationally intense part of an optimization algorithm. However, it has been noted that the convergence properties of second-order methods are far stronger than that of their first-order counterparts. Quasi-Newton methods, also called secant methods, are iterative algorithms which utilize gradient and function evaluations only, in order to construct successive approximations of the second-order information. Newton-like search direction can then be computed from the approximated Hessian. The mathematical principles of quasi-Newton methods are detailed in [Section 3.5.1](#).

Quasi-Newton methods were created in 1959 by Davidon [83]. Fletcher and Powell [108] modified Davidon's update to enforce symmetry of the estimate, a desired property of the Hessian approximation when used to define descent directions. Broyden [52] introduced a new class of methods in 1965, and a number of update formulas have since been developed, the most commonly used of which is the BFGS update, developed independently in 1970 by Broyden [53], Fletcher [106], Goldfarb [133] and Shanno [304]. The Symmetric Rank 1 (SR1) update does not enforce positive definiteness, and generates closer approximations to the Hessian when it is nonconvex [76]. Quasi-Newton methods have been demonstrated to have better convergence properties than other gradient-based methods such as conjugate gradients, and have been very

popular in the nonlinear optimization community [246, 145, 240, 354]. More information can be found in survey papers such as Ref. [86, 263, 266].

The use of quasi-Newton methods for the estimation of the trajectory second-order STMs in the MDDP algorithm is investigated in [Chapter 3](#) and [Chapter 4](#). Research has been conducted into multi-step quasi-Newton methods [113, 114], the use of which within the MDDP algorithm would constitute an interesting future study.

Trust-Region Methods Trust-region methods differ from step-length-based methods in that the magnitude of the step is predetermined, and the search direction depends on said magnitude. The idea is to restrict the quadratic approximation of the objective function to a trust region where the approximation is believed to be reliable. Then the minimum of the quadratic model can be accepted as the next iterate without a line-search. The size of the trust region, or trust-region radius, is updated at each iteration to better reflect the quality of the quadratic approximations. The method relies on the shifting of the Hessian matrix to ensure a descent direction and respect of the trust-region radius, and can therefore be used with nonconvex Hessians [78]. Trust-region methods were created by Levenberg [185] in 1944, who proposed a Hessian shifting algorithm for nonlinear least-square problems, and independently by Marquardt [199] in 1963, who also demonstrated the equivalence between Hessian shifting and step-length reduction. The technique was applied to general nonlinear programming by Goldfeldt et al. [134]. Powell [261] used the method in concert with a quasi-Newton approximation of the Hes-

sian, and demonstrated the convergence of his algorithm, as did Fletcher [107] for linearly constrained problems. Trust-region methods have been proved to have stronger local convergence properties than step-length-based methods [307, 330, 61]. The computation of exact trust-region step is addressed by More and Sorensen [216] [215], while inexact steps are the subject of Ref. [307]. Constrained trust-region methods have also been the subject of several studies [330, 259, 71], in particular when the constraints are simple bounds [22, 73, 74].

Because of the combined advantages of ensuring the quality of the quadratic approximation and producing positive-definite shifted Hessians, and a demonstrated utility in the spacecraft trajectory problem [340, 179], a trust-region algorithm is selected for the solution of the nonlinear subprograms in the MDDP algorithm. A thorough study of trust-region methods, as well as their applications, can be found in the monograph by Conn et al. [78].

Constrained Optimization The methods presented so far are all designed to solve unconstrained optimization problems, and have to be adapted in order to solve constrained problems. The theory of constrained optimization introduces Lagrange multipliers and the optimality conditions called Karush-Kuhn-Tucker or KKT conditions [172]. The theory is not presented here and can be found, alongside many practical considerations and details about the methods discussed hereafter, in optimization textbooks and survey papers such as Refs. [125, 127, 109, 129, 23, 230, 9, 194].

A number of methods have been devised to solve the KKT conditions for constrained programming. The first case usually treated is that of linear

equality constraints, since nonlinear constraints can be linearly approximated. Linear equality constrained problems are usually solved using generalized elimination methods, also called primal methods [9], so that the unconstrained techniques described above are used on a subspace of the decision variables, guaranteeing a decrease in the objective function while always satisfying the linear constraints. Such elimination methods include the projected gradient method, which projects the descent direction onto the set of feasible directions [287], and as been proved to be quadratically convergent when using quasi-Newton approximations to the Hessian [223]. The null-space method used in MDDP for the treatment of simple bounds, as well as the range-space technique used in the original HDDP algorithm for path constraints are variants of the projected gradient method [109]. The reduced gradient method decomposes the decision variables into a set of dependent and independent variables, optimizes the problem with respect to the independent variables only, and modifies the dependent variables accordingly, enforcing satisfaction of the constraints [347].

Linear approximations of the constraints are employed in a class of methods dubbed Sequential Quadratic Programming, or SQP methods. The nonlinear problem is reduced to a succession of quadratic programs, defined by quadratic approximations of the Lagrangian function of the original problem subject to linear approximations of the constraints. SQP methods directly solve for the KKT conditions by using a Newton or quasi-Newton method, and the method is sometimes called projected Lagrangian or Lagrange-Newton approach [229, 9]. SQP algorithms can use the powerful results of convex pro-

gramming to solve the quadratic subproblems [47]. SQP methods were first introduced by Wilson [346]. A number of authors, starting with Han [144] and Powell [263, 264, 266], studied the use of quasi-Newton approximations to the Hessian, using projected and reduced gradient methods to handle the linear constraints, and gave proofs of convergence. A survey of SQP methods can be found in Refs. [5, 126], their application to large-scale optimization problems discussed in Refs. [222, 297], and their performance is analyzed in Ref. [131]. Sequential Quadratic Programming techniques constitute an important class of methods for trajectory optimization, since their implementation in modern solvers such as KNITRO [62] and SNOPT [130] is commonly used by practitioners to solve challenging optimal control problems. For instance, the Jet Propulsion Laboratory (JPL) uses the MALTO program for preliminary trajectory design, which employs a direct multiple-shooting transcription and uses SNOPT for the optimization of the resulting NLP [309]. MALTO has been successfully used for the preliminary design of several missions, including that of the Jupiter Icy Moons Orbiter [343]. The direct pseudospectral collocation software GPOPS – III, the indirect collocation method OTIS, and the direct multiple-shooting software COPERNICUS [235, 236] all employ SNOPT, while the SOCS software includes its own implementation of the SQP method.

Primal methods for the linearly constrained problem like the projected-and reduced-gradient methods have been adapted to nonlinear constraints and coined “generalized reduced-gradient” (GRG) approaches [288, 1, 296]. However, primal approaches usually require that the sequence of iterates be feasible points, which is difficult to ensure with nonlinear constraints. A different class

of methods exist where satisfaction of the constraints of the original problem is not guaranteed until convergence. Penalty methods are approximate optimization methods which modify the objective function to include a penalty term when the constraints are not respected. The unconstrained minimization of the penalized objective function replaces the constrained problem, and the value of the penalty parameter dictates the quality of the approximation to the constraints. Courant [80] describes a quadratic penalty term which is still used in many methods today. A number of other penalty functions, using both the ℓ_1 and ℓ_2 norms, have been developed to handle nonlinear constraints, a survey of which can be found in Ref. [104]. The main disadvantage of penalty methods is that they usually require the penalty parameter to be raised to infinity in order to ensure constraints satisfaction, generating a well-documented ill-conditioning problem for the Hessian matrix [219, 220, 193].

When optimizing spacecraft trajectories, many of the constraints encountered are inequalities, which require special treatment in the algorithms. The three most common ways to handle inequalities are: 1) the addition of slack variables to the problem, which transform the inequality constraints into equalities, but add a large number of decision variables to the problem. The approach employed in the MDDP algorithm is based on these principles, but the slack variables are not explicitly computed. 2) The definition of an active-set strategy, which determines a subset of active constraints and treats them as equalities, while leaving the rest untouched, and is used in many general-purpose NLP solvers, such as SNOPT [130] and KNITRO [62]. The original HDDP algorithm employed an active-set quadratic programming algorithm

using a projected-gradient method for the treatment of path constraints [178].

3) The use of barrier or interior point methods[127]. Interior point methods [104] are a specific type of penalty method which enforces strict satisfaction of the inequality constraints at every iteration. The logarithmic barrier function introduced by Frisch [120] and the inverse barrier function of Carroll [67, 68] most commonly replace the objective functions in interior point methods. Like penalty methods, interior point techniques suffer from ill-conditioning of the problem when the penalty parameter is increased [219, 220]. However, since the mid-1980's and the publication of the polynomial-time linear programming algorithm of Karmarkar [162], interest in interior point methods has been rejuvenated and a number of algorithms have been developed [117, 214]. The large-scale optimization software IPOPT utilizes barrier functions for nonlinear interior point programming [336, 231, 337, 338], and KNITRO implements two different interior point algorithms. Interior point methods can also be used to solve the quadratic subproblem within SQP algorithms [40, 119].

Augmented Lagrangian (AL) methods were first developed independently by Powell [260] and Hestenes [150], as an answer to the ill-conditioning issues arising in pure penalty methods. The basic principle of the Augmented Lagrangian approach is to augment the original problem with a penalty term. The unconstrained minimization of the Lagrangian of the augmented problem replaces the constrained optimization. The addition of the Lagrangian term in the objective function means that the penalty parameter is not required to reach infinity anymore, and the ill-conditioning can be mitigated. From another point of view, a quadratic penalty term is added to the Lagrangian

in order to ensure its convexity. The mathematical concepts of AL methods as applied to equality and inequality constraints in the MDDP algorithm are presented in [Section 3.2](#). A thorough review of theoretical and practical properties of Augmented Lagrangian methods can be found in the books by Bertsekas [22] and Birgin and Martinez [34].

Augmented Lagrangian algorithms are based on the primal-dual philosophy of original Lagrangian methods, and solve an inner subproblem at each major iteration of the algorithm, before updating the multipliers in an outer loop. Therefore, the convergence of Augmented-Lagrangian-based solvers is dependent on both the inner loop convergence properties, and the chosen multipliers update formula [22, 194]. Tapia [321] presents a survey of possible multipliers update formulas and their history, and proposes a new update. The theoretical convergence of many variants of the Augmented Lagrangian methods has been extensively studied [209, 354, 72], as well as their use as merit functions within other algorithms such as SQP methods [299, 128, 112, 60] and DDP [191, 295]. Moreover, a number of authors used the notion of approximate convergence for the inner subproblem, and demonstrated that the inner loop does not need exact convergence for the Augmented Lagrangian algorithm to maintain its convergence properties, thus reducing the computational load of each inner loop subproblem [285, 255, 75]. Rockafellar [283, 284] introduced an implicit way to handle slack variables for inequality constrained problems. His approach is selected for the inequality constraints in the present work, as it has been demonstrated to be numerically superior to a number of other Augmented Lagrangian extensions to inequalities [35]. Several authors

mention the improved convergence properties of Augmented Lagrangian algorithms which leave the treatment of simple bounds or linear constraints to the inner loop solver [109, 75, 72]. The present work will leave simple control bounds to the null-space trust-region method used in the inner loop for the solution of the quadratic subprograms.

The Powell-Hestenes-Rockafellar[260, 150, 283, 258, 284] approach to the Augmented Lagrangian is implemented in MDDP, with dedicated multipliers update formulas for the path and terminal constraints. The method is general, does not necessitate the explicit inclusion of slack variables or of an active-set strategy, and permits the use of the previously developed unconstrained algorithm. Authors such as Bertsekas [22], Conn et al. [75] suggest that the use of Augmented Lagrangian techniques can be superior to the more common active-set strategy, especially for problems where the determination of an active-set is difficult, as can be the case with the path constraints of the trajectory optimization problem. Some of the algorithmic details such as the approximate convergence scheme are taken from Conn et al. [75], and are therefore similar to the implementation of the Augmented Lagrangian in the nonlinear solver LANCELOT [77]. MINOS [225] is another general-purpose NLP solver that uses a reduced-gradient approach for linearly constrained problems and a projected augmented Lagrangian approach for nonlinear constraints [224].

The transcription of optimal control problems into nonlinear programs often yields large dimensioned problems, with thousands or tens of thousands of parameters. Since NLP solvers rely heavily on the solutions to systems of

linear equations, most optimal control solvers see their computational complexity increase quadratically or cubically with the number of decision variables. However, most nonlinear programs generated by transcription are sparse problems, and the development of special methods for sparse optimization has greatly contributed to many advances in the trajectory optimization discipline [140, 30, 221, 119, 27, 137, 40]. Additionally, DDP methods are designed to limit the nonlinear growth in complexity with the number of time steps [217, 187].

1.2.1.4 Differential Dynamic Programming

Dynamic programming (DP) methods were introduced for the optimization of multi-stage decision processes. They are based on Bellman's Principle of Optimality, which states that:

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. (Bellman [11] Chap.III.3)

This principle permits to take advantage of the dynamical structure of the optimal control problem to decompose it into a succession of smaller subproblems at each instant in time, rather than solving a larger global problem. Bellman's Principle leads to the continuous Hamilton-Jacobi-Bellman (HJB) equations, a system of first-order nonlinear partial differential equations equivalent to Pontryagin's Minimum Principle [257], and their discrete counterpart,

the Bellman equation. Solving the HJB equation analytically or numerically yields the solution of the optimal control problem, which is effectively decomposed in a series of smaller subproblems, solved sequentially, in a process called backward induction. The subproblem corresponding to the last time period is solved first, and a control policy depending on the state is computed. The previous time period is solved, knowing that future decisions are made optimally, and the process is repeated until the first time period. Dynamic programming therefore allows the construction of an optimal feedback control law that accounts for arbitrary perturbations to the state [12]. This is particularly favorable in spacecraft trajectory optimization where force models are approximate and control policies have to be devised when the spacecraft is perturbed from the nominal trajectory. The main drawback of dynamic programming is the “curse of dimensionality”: dynamic programming solution methods generate a field of extremals, where the optimal control policies and objective value are known for any starting point, which requires large (often prohibitive) amounts of storage and computing power [56].

Differential DP consists in using quadratic approximations of the objective function around a reference trajectory and solving the resulting quadratic subproblems. The procedure was first introduced by Mayne [203], Jacobson and Mayne [157], and Gershwin and Jacobson [123], and has been the subject of many studies since, a survey of which was realized by Yakowitz [352]. The method finds control increments that improve the nominal trajectory locally and uses second-order information to obtain the optimal feedback control law. The new reference trajectory is then propagated forward using the control

law, and the whole process is iterated until convergence. DDP algorithms enjoy strong convergence properties, including quadratic convergence in a small neighborhood around the solution [218]. With only slight modifications, the method becomes equivalent to a stagewise implementation of Newton’s method [84]. DDP methods are close to direct methods, in that they parameterize the state and controls, and optimize the objective function directly. However, they have been shown to have a lower computational complexity than pure direct methods, since they solve a sequence of subproblems of smaller dimension. The complexity of the algorithm increases only linearly with the number of discretization points, whereas most nonlinear programs display quadratic or cubic increase [217, 189]. Moreover, Differential DP is not subject to the “curse of dimensionality” of traditional dynamic programming algorithms since the search space is restricted to a corridor around the nominal trajectory [350, 56]. This restriction implies however that the convergence of DDP algorithms can only be ensured locally. Finally, the application of Bellman’s Principle of Optimality and the creation of an optimal feedback law means that the conditions of optimality given by the calculus of variations are met, making DDP a hybrid direct/indirect method. Unlike classical indirect methods, the DDP approach has an increased radius of convergence, and is not as sensitive to poor initial guesses. DDP algorithms have been tested and used in a number of optimal control problems, and have demonstrated favorable convergence properties when compared to other large-scale optimization methods [349, 218, 350, 189]. The methods are used for applications as varied as water resources optimization problems (aqueduct design, reservoir operations analysis, water quality

maintenance) [349] or robotics problems such as the optimization of complex humanoid robots [322, 323]. The classic DDP approach is most effective for smooth unconstrained problems with convex Hessian matrices [188]. Unfortunately, almost no space trajectories yield convex Hessians, except when very near-optimal. Therefore, modifications must be made for practical use of DDP methods for the spacecraft trajectory optimization problem.

In this context, Whiffen developed a DDP algorithm dubbed Static / Dynamic Control Algorithm (SDC), which uses Hessian shifting in order to enforce convexity of the problem and penalty functions for constraints handling [340, 344]. SDC has been successfully implemented in the JPL’s Mystic software [342], and applied to the optimization of low-thrust spacecraft trajectories such as the Jupiter Icy Moons Orbiter [343], distant retrograde orbits at Europa [173], and the highly successful Dawn mission [278]. In efforts to further advance the development of DDP methods, Lantoine and Russell [174, 178, 179] recently extended the application of DDP to include an Augmented Lagrangian merit function for the treatment of terminal constraints, and to use STMs for sensitivity propagation. The use of Augmented Lagrangian has been investigated for both continuous and discrete DDP algorithms [70, 306, 295], and has been demonstrated to perform competitively with more traditional NLP techniques [191, 192, 351]. The main benefit of the STM approach is the decoupling of the optimization from the dynamics, allowing for 1) analytic solutions to the uncontrolled dynamics if available, 2) parallel computation, and 3) approximations to the expensive partial derivatives. The Hybrid Differential Dynamic Programming algorithm (HDDP) also

uses a range-space method for the treatment of path constraints, as well as a null-space trust-region method for the solution of the quadratic subproblems with simple bounds. HDDP has been successfully tested on a number of trajectories, including multi-revolution orbital transfer, and its results have been compared to solvers like IPOPT and SNOPT [179]. Lantoine and Russell [178] demonstrated that the sensitivities of the augmented objective function used in HDDP converged to the co-states of the optimal control problem, and could therefore be used as an initial guess for an indirect method. Moreover, the final solution found by HDDP is guaranteed to satisfy the Pontryagin Minimum Principle, and therefore to be optimal. While the HDDP algorithm is formulated using multiple phases, it is not a multiple-shooting algorithm, since the phases are solved successively and not independently.

1.2.1.5 Motivations for the Multiple-Shooting Differential Dynamic Programming Algorithm

A new DDP algorithm is developed in the present work. The good convergence properties of DDP methods, the generation of an optimal feedback control law, as well as the linear increase in computational effort when increasing the number of discretization steps makes DDP a good candidate for the optimization of long low-thrust spacecraft trajectories. The new MDDP algorithm is largely based on HDDP, and in particular uses STMs for sensitivity propagation. The DDP framework is used to solve a multiple-shooting transcription of the optimal control problem, since multiple-shooting formulations have been demonstrated to improve the robustness and convergence proper-

ties of both direct and indirect methods. The formulation enables a reduction of the sensitivity of each subproblem, improving the robustness and radius of convergence. Moreover, parallel implementation of the subproblems is possible, increasing the computational efficiency. The PHR Augmented Lagrangian approach is used for the treatment of path and terminal constraints, which allows a wide range of constraints to be handled in a general and flexible manner. The algorithm uses a trust-region method for the optimization of the nonlinear subprograms, and a null-space approach is used in the trust-region algorithm for the treatment of simple bounds on the decision variables. The algorithm is associated with an automated way to generate first- and second-order STMs and partial derivatives of the constraints and objective function using multi-complex step differentiation, which tremendously increases the versatility of the implementation. However, numerical evaluation of the second-order STMs constitutes the majority of the computational burden. Quasi-Newton methods are employed to estimate the second-order STMs, and demonstrated to drastically reduce the computational load.

1.2.2 Numerical Partial Derivatives

The computation of partial derivatives, or sensitivity analysis, is at the heart of most optimization and root-solving methods [127, 109, 27], and are essential to the MDDP algorithm developed in the present dissertation. The efficiency, convergence rate, and accuracy of derivatives-based algorithms depend directly on the quality of the computed partials: the theoretical convergence results generally assume that the functions and partials are sufficiently

smooth and continuous, and that exact derivatives can be obtained. However, computing exact derivatives is practically impossible, and accurate partials are almost always computationally expensive to obtain. In the case of trajectory optimization, the propagation of the trajectory and the computation of its sensitivities often are the heaviest computational load of the algorithm. Inaccurate derivatives generate sub-optimal steps for the search, yielding a higher number of iterations to solve the problem, or leading to algorithm divergence. These problems are accentuated when trajectories are highly nonlinear, as is the case for multiple-flyby orbits, multiple-body orbits, or multi-revolution spiral orbits.

The accuracy and efficiency of integration techniques for orbital motion have been the subject of numerous studies. The studies usually focus on comparing the accuracy of different integrators [118, 213, 142], or different regularizations and formulations of the equations of motion [332, 89, 210, 329]. The accuracy is commonly defined with respect to the true solution of the equations of motion, and a number of papers focus on finding the best way to estimate this accuracy [355, 356, 211, 20]. The computational time and the ease of implementation of the methods are other common comparison criteria [20, 181]. In Chapter 5, the focus is given to the accuracy of the partial derivatives of the trajectory with respect to its initial conditions. The trajectory is approximated by the process of numerical integration of the equations of motion, and this approximation and its sensitivities are ultimately needed as input to a root-solving or optimization algorithm. Therefore, the sensitivities to be used in gradient-based methods should be accurate derivatives of

the approximate trajectory. Unlike for navigation applications, the notion of a true trajectory and its partials does not play a role in the purest optimal control context. In fact, certain integrators that increase the accuracy of the trajectory propagation (with respect to the physical truth) can degrade the efficiency of an optimization or root-solving process, due to the presence of added noise in the gradients and high frequency dynamics [25].

Practitioners commonly use one of the following methods to obtain partial derivatives: 1) the propagation of the variational equations alongside the trajectory, 2) finite difference methods, or 3) automatic differentiation (AD). In the present study the complex-step derivative approximation (CSD) [311, 202, 181] is used in lieu of AD, as these two methods have been shown to be linked [201], and the CSD approach has gained traction recently due to its relative ease of implementation.

The propagation of the variational equations is typically done alongside the propagation of the trajectory, using the same integrator. The variational equations are added to the dynamics, and a new augmented state vector containing the state as well as the desired partials is propagated [3, 242]. This process requires the practitioner to have access to the derivatives of the equations of motion, which can be tedious or even impossible to obtain. In the present work, symbolic manipulation software (Maple) is utilized for the computation and implementation of the dynamics and the associated variational equations.

Finite difference methods [46] have been widely used for sensitivity

analysis, mainly due to their ease of implementation. When using finite differences, the following problem is introduced: it is well-known that finite differences formulas yield truncation errors and round-off errors [156, 115], which can destroy smoothness in the trailing digits and combine to form large errors in the sensitivities, especially when generating second- or high-order information, as is necessary for certain algorithms. High-order formulas can be generated for higher precision of the derivatives [116]. A method for automatic generation of high-order formulas for functions of multiple variables is explained and used in [Chapter 5](#). It is emphasized that high-order formulas imply using a large number of function evaluations, resulting in long computational times.

The CSD approximation was first introduced by the work of Lyness and Moler [197], and Lyness [196]. The modern formulation for first-order derivatives was given by Squire and Trapp [311], and its implementation described by Martins et al. [202]. Lantoine et al. [181] describe multi-complex variables that are used to obtain higher-order derivatives and their implementation. The present work uses complex and bi-complex variables to get the first- and second-order derivatives of the trajectory. A Fortran library overloading the operators and intrinsic procedures for bi-complex variables is implemented, as well as a Python preprocessor allowing for the automatic generation of complex and bi-complex code.

In this thesis, focus is given to second-order approximations of the trajectory, using both first- and second-order derivatives. As described in [Section 1.2.1.3](#), a number of methods only use first-order partials, but many

optimization and root-solving algorithms benefit greatly from accurate second-order information. Filters [345, 243, 198], root-solving methods (e.g. Halley's [121] and Laguerre's [239]), and optimization algorithms have been developed specifically to take advantage of second-order sensitivities. A number of methods have also been developed to approximate the second-order partials, as the exact partials usually are computationally expensive to obtain. Note that this second-order benefit does not matter in practice for very smooth, low-sensitivity problems, but it is critical for highly sensitive, highly nonlinear problems.

1.2.3 Numerical Propagation of Spacecraft Trajectories

The transcription methods introduced in Section 1.2.1.2 all necessitate the implicit or explicit integration of the dynamical equations of the optimal control problem. Therefore, the computational speed, robustness, and accuracy of the propagation methods are essential to the performance of the optimal control solver. The development of numerical methods for the integration of Ordinary Differential Equations (ODEs) has been studied extensively over the past decades. The typical goal when developing new integration schemes is to improve on both speed and accuracy; in this dissertation, the efficiency of an integrator is defined as its computational speed, assuming some reference level of accuracy is achieved. In the literature, number of different methods have been developed, including Runge-Kutta, multistep, symplectic, collocation, and Taylor series methods. The study and comparison of these methods is left to papers such as Refs. [355, 356, 118, 211, 213, 142, 20, 159], which

constitute excellent surveys of the different methods available to practitioners, as well as their caveats. In the present dissertation, particular attention is given to Taylor series methods, a recursive flavor of which is developed for the Kepler, Stark, and Circular Three-Body Problem (CRTBP). Montenbruck [212] demonstrated that the Taylor series methods have several advantages over other numerical schemes, including high and variable order and simple step control. Additionally, Taylor series methods naturally allow for the interpolation of states between time steps with no loss of accuracy [102].

The improvement of the accuracy and efficiency of a propagation method can result from a better numerical scheme, for instance the use of a higher-order Runge-Kutta integrator. A different method altogether consists in changing the formulation of the ODEs themselves, so that it can be exploited to achieve better integration performance, in some cases avoiding integration altogether. In the well-known case of the 2-body problem, an efficient scheme consists in using the Lagrange f and g functions, coupled with a solution to Kepler's equation using Universal Functions [7]. The so-called F and G series are a Taylor series representation of these Lagrange f and g functions. The advantage of such a representation lies in the fact that simple recursive formulations exist to obtain the coefficients of the F and G series [300, 44]. This method has been used to solve a variety of problems. Steffensen [313] first developed a method using recursive power series for the restricted 3-body problem. Rabe [268] adapted it for the case of periodic Trojan orbits, and Broucke [51] extended the method to solve the N-body problem. Fehlberg [102], Deprit and Price [87], and Broucke [50] also studied and used power series methods, which have

been shown to compare favorably to traditional numerical methods [88, 160].

Sconzo et al. [300] showed that the F and G Taylor series can be used efficiently as integrators, since their coefficients can be obtained to high order using a symbolic manipulator. Bond [44] developed a recursive formulation of the Lagrange coefficients for Keplerian motion. The generalized F and G series, extension of the classic F and G series to the N-body problem, were introduced by Papadakos [241], who also demonstrated their convergence. The F and G series integration method has also been used for orbit determination [14], for the propagation of certain cases of perturbed 2-body motion [310, 305], and for the solution to the Gauss problem (equivalent to Lambert's problem in the context of orbit determination, and therefore short times of flight) [6, pp. 251-258].

In this study, the F and G series method is extended to the so-called Stark problem, which corresponds to Keplerian motion perturbed by an inertially constant acceleration. Using the Stark problem to represent a low-thrust trajectory corresponds to a piecewise constant parameterization of the controls, which is adapted to preliminary design applications. Most implementations of low-thrust optimization algorithms use general ODE solvers instead of exploiting the custom Stark solution, resulting in higher propagation times. The computationally expensive nature of general ODE solutions has led some researchers to use reduced models, such as the Sims and Flanagan model [308] that employs a Kepler arc plus an impulsive ΔV to approximate a low-thrust arc. Such an approach avoids classic numerical integration and leads to significant runtime savings, but comes at the expense of accuracy.

The Sims-Flanagan model is used in software such as JPL’s MALTO [309] and Purdue University’s GALLOP [204]. Alternatively, Kirchgraber [168], Rufer [292], Poleshchikov [256] and Lantoine and Russell [176] give analytic solutions to the Stark problem, leading to more accurate solutions, for the case of two-body plus piecewise constant thrust. However, analytic techniques suffer from complex formulations and implementations, may include various levels of approximation, and require transcendental functions such as elliptic integrals to solve. It is noted that Yam et al. [353] make a similar argument, and suggests the use of the Modern Taylor Series method to propagate the Stark problem as an approximation to low-thrust trajectories, instead of standard numerical integration.

The F and G approach is also applied to the 3D CRTBP. Automated trajectory design methods stand to benefit from using three-body dynamics, which provide natural opportunities for low-energy transfers [180]. Moreover, Jorba and Zou [160] demonstrated that their variable-order Taylor series solution to the CRTBP had performance comparable to `dop853`, an explicit variable-order Runge-Kutta integrator of order 8, an additional motivation for the development of a Taylor series solution.

Additionally, both the present work and Yam et al. [353] make use of the Sundman transformation in order to generate an efficient geometric discretization scheme. For the Kepler and Stark problems, the classic transformation, employed in Yam et al. [353], is extended to the generalized Sundman transformations. The original transformation was introduced by Karl Sundman as a way to regularize the equations of motion of the 3-body problem and avoid

collision singularities [316]. The use of the transformation has proven to be a remarkably simple way to obtain efficient discretization schemes, especially for highly eccentric orbits [19, 353]. The Sundman transformation provides a near-equalization of the truncation error at each integration step [332] and often reduces the global truncation error by reducing dynamical instability [332, 8]. Both of these effects result in an improved integration performance, because the lower global truncation error allows the integrator to take larger steps. Furthermore, the equalization of the local truncation error makes the use of fixed-step integrators practical, noting that these are often more efficient than their variable-step counterpart [226]. Szebehely [319] extensively describes and justifies the need for regularization methods in the CRTBP, as close approaches between bodies are common and lead to singularities in the limit of collisions. In particular, Szebehely describes regularization techniques by Birkhoff [36], Lemaitre [184], and Thiele [324] and Burrau [59], and gives a historical survey of regularization methods. However, in the current study, only basic modifications to the Sundman transformation for the CRTBP will be considered. The Sundman transformation and its application to both the Kepler and CRTB problems are detailed in [Section 2.2.3](#), while the F and G solutions to these problems is developed in [Chapter 6](#).

1.3 Outline of the Dissertation

The present dissertation consists of four main chapters detailing the main contributions of the research, and additional supporting material. [Chapter 2](#) introduces some mathematical notions required for the understanding of

subsequent chapters. Some mathematical notations and definitions are presented, alongside the definition of State Transition Matrices (STMs) and of the main dynamical models used for the approximation of low-thrust trajectories, the Kepler, Stark and CRTB problems. The Sundman transformation and its application to the aforementioned dynamical models are introduced.

The theoretical development of the Multiple-Shooting Differential Dynamic Programming (MDDP) algorithm is given in [Chapter 3](#). The general multiple-phase optimal control problem is transcribed according to multiple-shooting principles, and the adopted Augmented Lagrangian strategy is described. The quadratic expansions and update equations necessary to the solution of the multiple-shooting problem are detailed, and the implementation of the MDDP algorithm is summarized. Finally, the application of quasi-Newton principles to the computation of second-order STMs is presented.

[Chapter 4](#) contains numerical solutions of diverse optimal control problems obtained using the MDDP algorithm. The effects of the multiple-shooting formulation are investigated, and the efficiency of the quasi-Newton algorithm is analyzed. [Chapter 4](#) also contains a description of the set of application cases used for the performance evaluations.

The efficiency and robustness of the MDDP algorithm and of other derivatives-based optimization methods is heavily influenced by the quality of the partial computation methods. Three commonly employed numerical methods for computing the STMs are described and compared in [Chapter 5](#).

In [Chapter 6](#), the F and G series solutions to the Stark and CRTB prob-

lems are developed. The Sundman transformation's effects are analyzed, and the new methods are compared to traditional numerical integration methods.

Finally, [Chapter 7](#) summarizes the main findings of the dissertation, and discusses potential future research. Appendices contain additional information and algorithms, as well as a list of the publications related to this research.

1.4 Summary of Contributions

Several contributions to the field of numerical optimal control and low-thrust trajectory optimization are put forth in the present dissertation. These advances have been the subject of publications which are listed in [Chapter D](#).

- A DDP algorithm using an Augmented Lagrangian approach for the treatment of path and terminal equality and inequality constraints is developed. Simple bounds on the decision variables are treated using a null-space trust-region method.
- A robust null-space trust-region algorithm is developed. The MDDP algorithm depends on a null-space trust-region for the solution of simply bounded nonlinear subprograms. A number of heuristic safeguards are developed to ensure robustness of this critical inner-loop solver.
- The first multiple-shooting formulation of a DDP algorithm is presented, which aims at improving the robustness and computational efficiency of DDP methods, as well as increasing the potential for parallel implementation. The effects of the multiple-shooting transcription are analyzed.

- The use of quasi-Newton methods for second-order State-Transition Matrices estimation within a DDP algorithm is investigated, and significant improvement of the computational efficiency is demonstrated.
- Recursive Taylor Series solutions to the Stark and Circular Restricted Three-Body problems are developed that seek to 1) preserve the accuracy of the analytic solutions when available, 2) be simple to implement, 3) avoid classic numerical integration, and 4) be computationally efficient. The F and G approach is compared to traditional numerical integration methods. The use of the Sundman transformation is investigated.
- Three commonly used methods of computation of the State-Transition Matrices are compared, and previously unpublished subtleties are revealed. Particular attention is given to the caveats of variable-step integration for sensitivity computation, and recommendations are made for the practitioner.

The contributions listed above are supported by additional work. A Fortran dynamic library is developed to define bicomplex numbers and overload all intrinsic operators. A text manipulation tool called `generize`, implemented using Python, generates complex and bicomplex version of user inputted Fortran subroutines. The tool is used to make generic versions Fortran codes, which can then be called using real, complex or bicomplex numbers, and permits a fast implementation of multicomplex step differentiation. Bicomplex differentiation is used in all applications of the MDDP algorithm, except where noted otherwise. The use of an automated differentiation method

such as multi-complex differentiation increases the flexibility of the algorithm because the dynamics and constraints can be changed without having to derive partial derivatives. A generic finite differences code is created for the comparison of finite differences methods. Finally, the MDDP algorithm is implemented in Fortran 2003, compiled using the Intel Visual Fortran compiler XE 12.1.6 on a Windows 7 workstation. Careful attention is given to the efficiency of all numerical algorithms and their implementation, and the software is designed to be modular and user friendly. In particular, dynamic libraries are used to define the optimal control problems independently of the main solver, allowing for the rapid and flexible implementation of new problems.

Chapter 2

Mathematical Background

The present chapter serves as an introduction to some of the mathematical concepts used in the subsequent material. Common notations and definitions are presented, as well as classical equations of motion (EOMs) used in trajectory design.

2.1 Notations and Conventions

In the remainder of this document, scalars are denoted in plain text, and vectors in boldface:

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \dots \\ X_N \end{bmatrix} \quad (2.1)$$

In general, the \mathbf{X} vector is the state vector, while \mathbf{r} and \mathbf{v} indicate position and velocity vectors, respectively. Matrices and tensors are in plain text. A colon is used to denote all of the elements in a particular dimension of an array, in a way similar to the MATLAB notation: $M(i, :)$ represents the i^{th} row of the M matrix, $M(:, j)$ its j^{th} column, etc.

All matrix calculus will be noted using the numerator-layout convention and derivation with respect to a scalar or a vector is usually indicated with a subscript, such that, if J is a scalar function of \mathbf{X} , its gradient and Hessian

are defined as:

$$J_X = \frac{\partial J}{\partial \mathbf{X}} = \left[\frac{\partial J}{\partial X_1} \frac{\partial J}{\partial X_2} \cdots \frac{\partial J}{\partial X_N} \right] \quad (2.2)$$

$$J_{XX} = \frac{\partial^2 J}{\partial \mathbf{X}^2} = \frac{\partial}{\partial \mathbf{X}} \left[\frac{\partial J}{\partial \mathbf{X}} \right]^T = \begin{bmatrix} \frac{\partial J_{X_1}}{\partial X_1} & \frac{\partial J_{X_1}}{\partial X_2} & \cdots & \frac{\partial J_{X_1}}{\partial X_N} \\ \frac{\partial J_{X_2}}{\partial X_1} & \frac{\partial J_{X_2}}{\partial X_2} & \cdots & \frac{\partial J_{X_2}}{\partial X_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial J_{X_N}}{\partial X_1} & \frac{\partial J_{X_N}}{\partial X_2} & \cdots & \frac{\partial J_{X_N}}{\partial X_N} \end{bmatrix} \quad (2.3)$$

For a vector-valued function $\mathbf{f}(\mathbf{X})$, the Jacobian and Hessian matrices are defined such that:

$$\mathbf{f}_X(i, j) = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{X}} \right)(i, j) = \frac{\partial f_i}{\partial X_j} \quad (2.4)$$

$$\mathbf{f}_{XX}(i, j, k) = \left(\frac{\partial^2 \mathbf{f}}{\partial \mathbf{X}^2} \right)(i, j, k) = \frac{\partial^2 f_i}{\partial X_j \partial X_k} \quad (2.5)$$

The functions considered in this dissertation are assumed to have continuous second-order derivatives, and therefore Schwarz's theorem can be used to conclude that $J_X X$ and every $\frac{\partial^2 f_i}{\partial \mathbf{X}^2}$ are symmetric matrices. Since the second-order partials of a vector-valued function with respect to another vector is a tensor of dimension 3, a number of tensor operations have to be introduced. They will be noted using the \bullet_m operator, called the m -mode tensor product and defined as:

$$(\mathbf{X} \bullet_1 T)(i, j) = \sum_{p=1}^N T(p, i, j) X(p) \quad (2.6)$$

$$(\mathbf{X} \bullet_2 T)(i, j) = \sum_{p=1}^N T(i, p, j) X(p) \quad (2.7)$$

$$(\mathbf{X} \bullet_3 T)(i, j) = \sum_{p=1}^N T(i, j, p) X(p) \quad (2.8)$$

$$(M \bullet_1 T)(i, j, k) = \sum_{p=1}^N T(p, j, k) M(i, p) \quad (2.9)$$

$$(M \bullet_2 T)(i, j, k) = \sum_{p=1}^N T(j, p, k) M(j, p) \quad (2.10)$$

$$(M \bullet_3 T)(i, j, k) = \sum_{p=1}^N T(j, k, p) M(k, p) \quad (2.11)$$

$$(M \bullet T \bullet M)(i, j, k) = \sum_{p=1}^N \sum_{q=1}^N T(i, p, q) M(p, j) M(q, k) \quad (2.12)$$

where \mathbf{X} is an $N \times 1$ vector, M is a $N \times N$ matrix, and T an $N \times N \times N$ tensor.

In results sections and for example problems, units are usually normalized, and noted DU (Distance Unit) or LU (Length Unit), MU (Mass Unit), and TU (Time Unit).

2.1.1 State-Transition Matrices

In this work, specific attentions is given to the first- and second-order State-Transition Matrices (STMs), since they are used to capture the sensitivity of a trajectory with respect to its initial conditions and controls [242, 161, 327]. The first- and second-order STMs from time t_0 to t are noted $\Phi^1(t, t_0)$ and $\Phi^2(t, t_0)$ and defined as the first- and second-order partial derivative of the state $\mathbf{X}(t)$ with respect to the initial state $\mathbf{X}_0 = \mathbf{X}(t_0)$:

$$\Phi^1(t, t_0) = \frac{\partial \mathbf{X}(t)}{\partial \mathbf{X}_0} \quad (2.13)$$

$$\Phi^2(t, t_0) = \frac{\partial^2 \mathbf{X}(t)}{\partial \mathbf{X}_0^2} \quad (2.14)$$

such that:

$$\delta \mathbf{X}(t) = \Phi^1(t, t_0) \delta \mathbf{X}_0 + \frac{1}{2} \delta \mathbf{X}_0^T \bullet_2 \Phi^2(t, t_0) \delta \mathbf{X}_0 + \mathcal{O}(\delta X_0^3) \quad (2.15)$$

The definition of Φ^2 in Eq. (2.14) yields, for any (i, j, k) , using Schwarz's theorem and assuming that the second-order derivatives are continuous:

$$\Phi^2(t, t_0)_{i,j,k} = \frac{\partial^2 \mathbf{X}_i(t)}{\partial \mathbf{X}_{0,j} \partial \mathbf{X}_{0,k}} = \frac{\partial^2 \mathbf{X}_i(t)}{\partial \mathbf{X}_{0,k} \partial \mathbf{X}_{0,j}} = \Phi^2(t, t_0)_{i,k,j} \quad (2.16)$$

This symmetry can be exploited when computing the partials. In the remainder of this dissertation, when first- and second-order STMs are needed, they are computed using one of the three methods presented in Section 5.1. Unless indicated otherwise, the multi-complex step derivatives are used.

2.2 Dynamical Models

This section presents three of the most commonly used dynamical systems for trajectory optimization, as well as the application of the Sundman transformation to these systems.

2.2.1 Two-Body Problem

The two-body problem, or Kepler problem, represents the motion of a massless body (e.g. a spacecraft) through the gravitational field of a point mass. This system has been studied for centuries, and its properties can be found in a number of astrodynamics textbooks, such as Bate et al. [6]. The two-body problem is one of the only models used in astrodynamics for which an analytical solution exists for the trajectory and corresponding first-

and second-order STMs. The improvement in computational time resulting from using an analytical solution rather than numerical integration makes the Kepler problem an attractive approximation for preliminary trajectory design. For this purpose, Sims and Flanagan [308] introduce the Sims-Flanagan model, which employs Keplerian arcs to link impulsive $\Delta\mathbf{v}$'s. The Sims-Flanagan approach is successfully implemented in low-thrust optimization software such as MALTO, developed at the Jet Propulsion Laboratory [309], and GALLOP, developed at Purdue University [204].

The equations of motion of the Kepler problem in time are:

$$\dot{\mathbf{X}} = \frac{d\mathbf{X}}{dt} = \begin{bmatrix} \mathbf{v} \\ -\frac{\mu}{r^3}\mathbf{r} \end{bmatrix} \quad (2.17)$$

2.2.1.1 Stark Problem

The so-called Stark problem is a perturbed 2-body problem, where the perturbation is inertially constant in magnitude and direction. The German physicist Johannes Stark, the namesake of the problem, discovered the shifting and splitting of spectral lines of atoms and molecules in the presence of an external static electric field. This phenomenon is now commonly known as Stark effect [312]. Apart from its importance in nuclear physics, the Stark effect can also approximate the behavior of many astrodynamical systems. A historical survey of the extensive research conducted on solving the Stark problem's equations of motion can be found in Lantoine and Russell [176]. Examples of the Stark effect in celestial mechanics include the propagation of any system with an inertially constant or approximately constant perturbation,

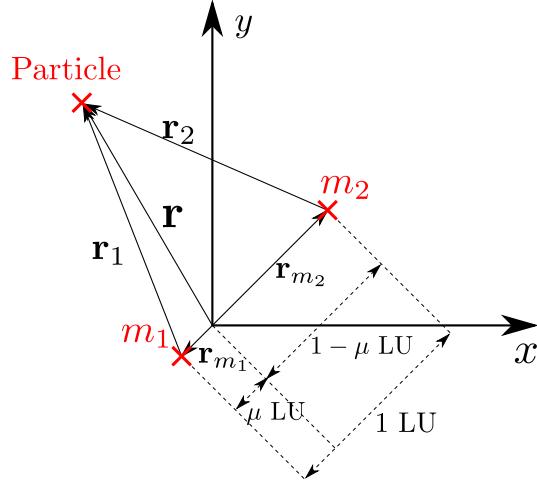


Figure 2.1: Sketch of the inertial frame CRTBP

such as a finite thrust arc, solar radiation pressure, or the direct influence of a distant third body. Examples of use include Namouni and Guzzo [228], Pástor [244]. Time varying perturbations can be approximated by the Stark model with the use of sufficiently small time steps [353, 357], making it a good candidate for the approximation of low-thrust trajectories. In the remainder of this dissertation, unless indicated otherwise, two-body low-thrust trajectories are approximated using the Stark model.

The equations of motion of the Stark problem in time are:

$$\dot{\mathbf{X}} = \frac{d\mathbf{X}}{dt} = \begin{bmatrix} \mathbf{v} \\ -\frac{\mu}{r^3} \mathbf{r} + \mathbf{p} \end{bmatrix} \quad (2.18)$$

where \mathbf{p} is the perturbing acceleration.

2.2.2 Circular Restricted Three-Body Problem (CRTBP)

The Circular Restricted Three-Body Problem (CRTBP) is a well-known astrodynamics problem, useful for a variety of applications, from modeling the Earth-Moon system to the exploitation of invariant manifolds for satellite tour design [136, 2, 177]. The CRTBP model consists of a massless particle moving through the gravitational field created by two point-masses in circular orbit around each other. An extensive description of the CRTBP can be found in the seminal text by Szebehely [319].

For the derivation of the F and G CRTBP series, which is the subject of [Chapter 6](#), the equations of motion are expressed in the inertial frame. The inertial frame formulation is preferred in the context of obtaining the recursion equations. Moreover, many advanced design tools, such as Mystic, express the CRTBP in the inertial frame [342]. In the inertial formulation, explicit knowledge of the states of the two masses is also necessary. The augmented state vector, including all three bodies, is formed as:

$$\mathbf{X} = [\mathbf{r}^T \ \mathbf{v}^T \ \mathbf{r}_{m_1}^T \ \mathbf{v}_{m_1}^T \ \mathbf{r}_{m_2}^T \ \mathbf{v}_{m_2}^T]^T \quad (2.19)$$

where \mathbf{r} , \mathbf{v} are the position and velocity of the massless particle in an inertial frame centered at the barycenter of the system; \mathbf{r}_{m_i} , \mathbf{v}_{m_i} are the position and velocity of m_i , where m_1 is the primary (the heavier mass) and m_2 the

secondary (the lighter mass, see Fig. 2.1).

$$\dot{\mathbf{X}} = \frac{d\mathbf{X}}{dt} = \begin{bmatrix} \mathbf{v} \\ -\frac{Gm_1}{r_1^3}\mathbf{r}_1 - \frac{Gm_2}{r_2^3}\mathbf{r}_2 \\ -\frac{\mathbf{v}_{m_1}}{r_{m_1}} \\ -\frac{\mathbf{v}_{m_2}}{r_{m_2}} \end{bmatrix} \quad (2.20)$$

By definition (see Fig. 2.1):

$$\mathbf{r}_1 = \mathbf{r} - \mathbf{r}_{m_1} \quad (2.21)$$

$$\mathbf{r}_2 = \mathbf{r} - \mathbf{r}_{m_2} \quad (2.22)$$

$$\mathbf{r}_{m_2} = -\frac{r_{m_2}}{r_{m_1}}\mathbf{r}_{m_1} = -c\mathbf{r}_{m_1} \quad (2.23)$$

$$\mathbf{v}_{m_2} = -c\mathbf{v}_{m_1} \quad (2.24)$$

with $c = \frac{r_{m_2}}{r_{m_1}}$.

The units are all normalized as shown in Eqs. (2.25)-(2.27):

$$Gm_1 + Gm_2 \equiv 1 \text{ LU}^3/\text{TU}^2 \quad (2.25)$$

$$r_{m_1} + r_{m_2} \equiv 1 \text{ LU} \quad (2.26)$$

$$\omega \equiv 1 \text{ rad/TU} \quad (2.27)$$

where $\omega = \sqrt{\frac{G(m_1+m_2)}{r_{12}^3}}$ is the system's angular velocity. With $\mu = \frac{m_2}{m_1+m_2}$,

those relations become:

$$Gm_1 = 1 - \mu \text{ LU}^3/\text{TU}^2 \quad (2.28)$$

$$Gm_2 = \mu \text{ LU}^3/\text{TU}^2 \quad (2.29)$$

$$r_{m_1} = \mu \text{ LU} \quad (2.30)$$

$$r_{m_2} = 1 - \mu \text{ LU} \quad (2.31)$$

$$\omega = 1 \text{ rad/TU} \quad (2.32)$$

Using Eqs. (2.21)-(2.32), the location of the secondary can be expressed in terms of the primary, and the state vector can be simplified: $\mathbf{X} = [\mathbf{r}^T \mathbf{v}^T \mathbf{r}_{m_1}^T \mathbf{v}_{m_1}^T]^T$.

The equations of motion become:

$$\dot{\mathbf{X}} = \begin{bmatrix} \mathbf{v} \\ p\mathbf{r} + q\mathbf{r}_{m_1} \\ \mathbf{v}_{m_1} \\ -\mathbf{r}_{m_1} \end{bmatrix} \text{ with } \begin{cases} p = -\frac{1-\mu}{r_1^3} - \frac{\mu}{r_2^3} \\ q = \frac{1-\mu}{r_1^3} - \frac{1-\mu}{r_2^3} \end{cases} \quad (2.33)$$

In the case of the Circular RTBP, the two masses m_1 and m_2 are in circular orbits around each other. Position and velocity of m_1 , when necessary, are simply computed by rotation:

$$\begin{aligned} \mathbf{r}_{m_1}(t) &= \begin{bmatrix} \cos(t - t_0) & -\sin(t - t_0) & 0 \\ \sin(t - t_0) & \cos(t - t_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{r}_{m_1,0} \\ \mathbf{v}_{m_1}(t) &= \begin{bmatrix} \cos(t - t_0) & -\sin(t - t_0) & 0 \\ \sin(t - t_0) & \cos(t - t_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{v}_{m_1,0} \end{aligned} \quad (2.34)$$

Finally, the equations of motion to be solved are:

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{r}} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ p\mathbf{r} + q\mathbf{r}_{m_1} \end{bmatrix} \quad (2.35)$$

2.2.3 The Generalized Sundman Transformation

In this section, the generalized Sundman transformation and its application to astrodynamics are briefly surveyed, and the effects of the transformation on the discretization of the orbit are demonstrated.

2.2.3.1 Presentation and Historical Survey

The classic Sundman transformation was introduced in 1912 by Karl F. Sundman, as a way to regularize the equations of motion and to avoid collision singularities in the 3-body problem [316]. It consists of a transformation of the independent variable from time t to τ , given by:

$$dt = r \, d\tau \quad (2.36)$$

where r is the magnitude of the radius vector and τ is the new independent variable.

It is well known that the use of the Sundman transformation improves integration performance by reducing and uniformizing the local truncation error [318]. The numerical behavior of the regularized expressions becomes less dependent on the eccentricity when compared to the use of time as the independent variable [314]. The transformation was later extended to the so-called generalized Sundman transformation:

$$dt = cr^\alpha \, d\tau \quad (2.37)$$

where c is a constant and α is a positive constant.

Many authors have studied the generalized Sundman transformation in order to find the optimal value for α and to understand why it presents such numerical benefits. Baumgarte [8] showed that the transformation with $\alpha = 1$ reduces dynamical instability in the Lyapunov sense. Velez [332] tied this dynamical instability to error propagation, and further stated that the instability can be removed entirely by using $\alpha = 2$. Bond [45] showed that this $\alpha = 2$ case corresponds to transforming the equations of motion to oscillator form. However, Velez also explained that increasing α leads to higher numerical instability, and could therefore be detrimental to the overall accuracy of the propagation. Velez [332], Feagin and Mikkilineni [101], and Nacozy [226] all show that in order to reduce the local truncation error, there is not a single optimal transformation for all problems, or even for all points on one orbit. The ideal value for α depends on the problem, on the location on the orbit and on the order of the integrator. However, Merson [207], Velez [332] and Nacozy [226] all agree that the case $\alpha = 3/2$ is the best choice for most satellite applications, because they found it most efficient in the case of a J_2 perturbation. It is well known that $\alpha = 1$ and $\alpha = 2$ make the independent variable τ proportional to eccentric and true anomalies, respectively. Because the $\alpha = 3/2$ case proved ideal for heavily used scenarios, Nacozy [227] introduced the *intermediate anomaly*, corresponding to $\alpha = 3/2$.

Authors have also proposed the idea of a varying α [226], or more generally of a transformation of type:

$$dt = g(r) d\tau \quad (2.38)$$

where g is an arbitrary function of r [317, 103]. However, this more complicated type of transformation will not be considered in this document.

An important point, made by most of the cited authors, is that the Sundman transformation and its extensions introduce an extra differential equation in the problem: it is now necessary to integrate Eq. (2.37) in order to keep track of time. In fact, this new differential equation exhibits part or all of the dynamic instability that was removed from the equations of motion. However, if the equations of motion are autonomous, then there is no feedback of the instability into the coordinates, and the benefits of the transformation usually overcome this disadvantage.

2.2.3.2 Application of the Sundman Transformation to the Kepler and Stark Problems

In the present work, when applying the Sundman transformation to the Kepler and Stark problems, a time transformation of type $dt = cr^\alpha d\tau$, with $c = 1$ and α constant is studied.

When using the transformation, the independent variable is changed from time to τ , so that the traditional ODE:

$$\dot{\mathbf{X}} = \frac{d\mathbf{X}}{dt} = \mathbf{f}(t, \mathbf{X}) \quad (2.39)$$

becomes:

$$\begin{cases} \mathbf{X}' = \frac{d\mathbf{X}}{d\tau} = \frac{d\mathbf{X}}{dt} \frac{dt}{d\tau} = cr^\alpha \mathbf{f}(t, \mathbf{X}) \\ \frac{dt}{d\tau} = cr^\alpha \end{cases} \quad (2.40)$$

where $(x)'$ and (\dot{x}) designate differentiations of the dummy variable x with respect to τ and time, respectively.

It is important to discuss the case of second-order ODEs, as they are heavily used in astrodynamics. The conventional way of handling a second-order ODE is to create a state vector containing the state and its first derivative, and effectively converting the second-order equation to a system of first-order equations. For astrodynamics applications, the typical state vector contains position and velocity:

$$\mathbf{X} = \begin{bmatrix} \mathbf{r} \\ \frac{d\mathbf{r}}{dt} = \mathbf{v} \end{bmatrix} \quad (2.41)$$

When using the Sundman transformation with a first order integrator, two formulations are possible. In the first formulation, the state vector is the one presented in Eq. (2.41), and the propagation essentially corresponds to a first-order integration of the position and velocity (\mathbf{r} and \mathbf{v}). For an arbitrary time-acceleration \mathbf{a} :

$$\mathbf{X}' = \begin{bmatrix} \mathbf{r}' \\ \mathbf{v}' \end{bmatrix} = \begin{bmatrix} cr^\alpha \mathbf{v} \\ cr^\alpha \mathbf{a} \end{bmatrix} \quad (2.42)$$

Equation (2.42) will be called “time-velocity” equations of motion.

In the second formulation, the following state vector is used:

$$\mathbf{X} = \begin{bmatrix} \mathbf{r} \\ \mathbf{r}' \end{bmatrix} \quad (2.43)$$

where the integration now corresponds to solving a second-order ODE in τ , yielding:

$$\begin{aligned} \mathbf{X}' &= \begin{bmatrix} \mathbf{r}' \\ \mathbf{r}'' \end{bmatrix} = \begin{bmatrix} \mathbf{r}' \\ (cr^\alpha)^2 \mathbf{a} + (cr^\alpha)' \mathbf{v} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{r}' \\ (cr^\alpha)^2 \mathbf{a} + \frac{\alpha}{r^2} (\mathbf{r} \cdot \mathbf{r}') \mathbf{r}' \end{bmatrix} \end{aligned} \quad (2.44)$$

since

$$\mathbf{r}'' = \frac{d}{d\tau} \left(\frac{d\mathbf{r}}{d\tau} \right) = \frac{d}{d\tau} (cr^\alpha \mathbf{v}) = (cr^\alpha)^2 \mathbf{a} + (cr^\alpha)' \mathbf{v} \quad (2.45)$$

and

$$(cr^\alpha)' \mathbf{v} = \alpha cr^{\alpha-1} r' \mathbf{v} = \alpha r' \frac{r}{r} \frac{cr^\alpha}{r} \mathbf{v} = \frac{\alpha}{r^2} rr' \mathbf{r}' \quad (2.46)$$

The true velocity of the particle is obtained from the τ -velocity:

$$\mathbf{v} = \frac{1}{cr^\alpha} \mathbf{r}' \quad (2.47)$$

Equation (2.44) will be called “ τ -velocity” equations of motion. It is shown in Section 6.2.1 that the Taylor series methods correspond to a τ -velocity method and associated state from Eq. (2.43). For the benchmark Runge-Kutta integrator used in Section 6.3.1, results from both formulations are presented.

Figure 2.2, similar to that found in Berry and Healy [19], presents the result of high accuracy integrations of Eq. (2.40) using 2-body dynamics, for different values of α . The output nodes are equally spaced in τ , although $\Delta\tau$ is not the same from one trajectory to another: the τ -period (τ_p) depends on the value of α , and $\Delta\tau = \tau_p/50$. Table 2.1 gives the formulas for the computation of τ_p . Details on the formulas are given in Appendix A. Figure 2.3 presents the τ -period as a function of a or e for those four cases. For $\alpha = 0$ or 1, the period depends solely on a , for $\alpha = 3/2$, it depends solely on e , and finally for $\alpha = 2$, it depends on both a and e . For visualization purposes, a is fixed at 1 when varying e , and e is fixed at 0.7 when varying a .

The typical clustering of points around apoapse is easily seen in Fig. 2.2 for the time case ($\alpha = 0$), and the gradual shifting of that clustering changes

α	τ proportional to	τ_p
0	Time	$\frac{2\pi}{nc}$
1	Eccentric Anomaly	$\frac{2\pi}{nca}$
3/2	Intermediate Anomaly	$\frac{4 K(\sqrt{\frac{2e}{1+e}})}{c\sqrt{\mu(1+e)}}$
2	True Anomaly	$\frac{2\pi}{nc\sqrt{a(1-e^2)}}$

$n = \sqrt{\frac{\mu}{a^3}}$ is the mean orbital motion
 $K(\cdot)$ is the complete elliptic integral of the first kind
 μ is the standard gravitational parameter
 See Appendix A for details

Table 2.1: τ -period for different values of α

when increasing α . A value of $\alpha = 1$ corresponds to a regular geometric spacing, proportional to the eccentric anomaly, and independent of whether the particle is close to apoapse or periapse. The points start to cluster around periapse for $\alpha = 3/2$, and the effect is exaggerated further for $\alpha = 2$. Note that the discretization in the case of $\alpha = 3/2$ most closely mimics the behavior of conventional variable-step numerical integrators, when compared to the other discretizations of Fig. 2.2. Therefore, the Sundman transformations, in particular the $\alpha = 3/2$ case, are often used in fixed-step integrators. The discretizations shown in Fig. 2.2 agree with the conclusions of many authors, in particular those of Berry and Healy [19].

In the remainder of this document, particular attention is given to the

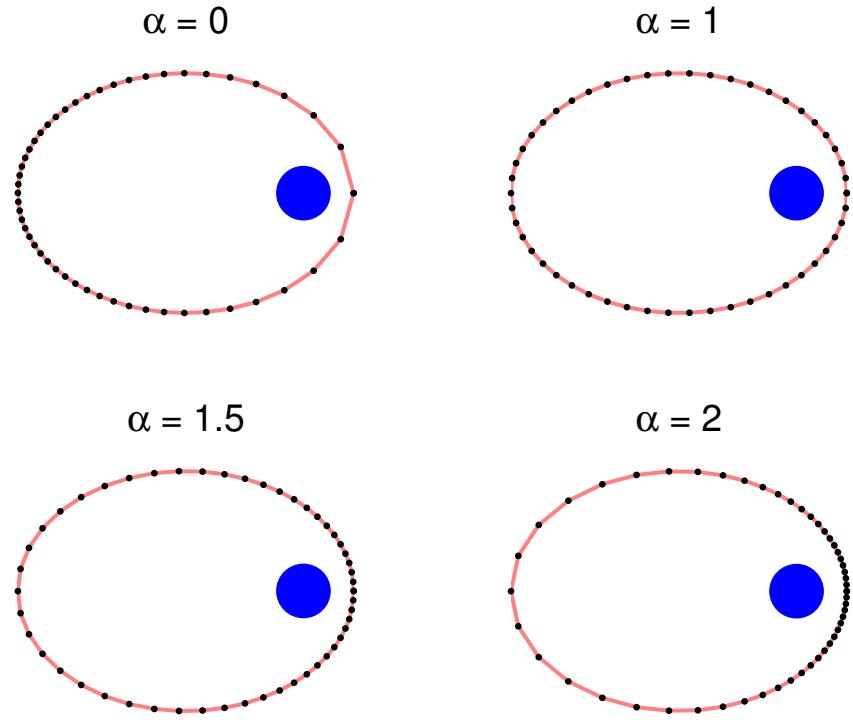


Figure 2.2: Effect of the Sundman power law. $a = 1$, $e = 0.7$

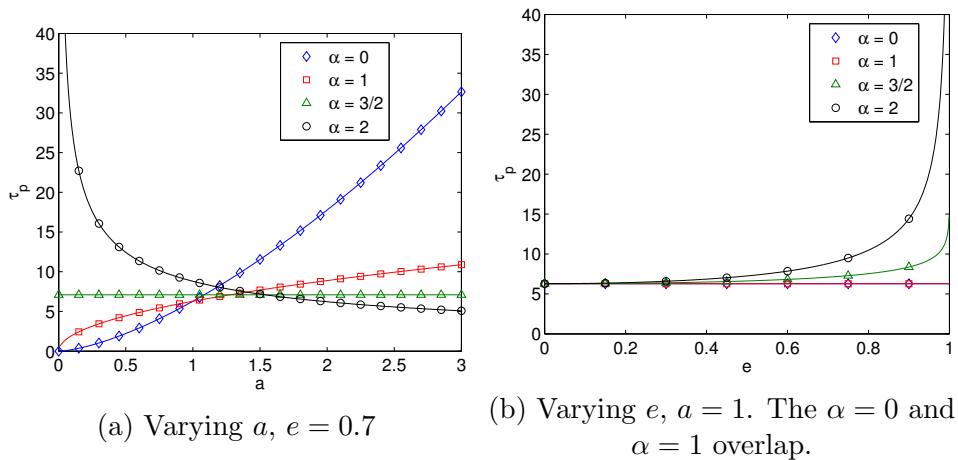


Figure 2.3: τ -period vs. orbital elements

eccentric anomaly case ($\alpha = 1$) because the regular geometric discretization is ideally suited for modeling a perturbation or optimizing a control. In [Section 6.3.1.3](#), the distribution of the local error along one revolution is presented for all the propagation methods and for different values of α . It is shown that the optimal α is strongly dependent on the integrator. In particular, the optimal α for Taylor series methods is shown to be 1, both a beneficial and serendipitous result.

2.2.3.3 Application to the CRTBP

The Sundman transformation has been proved to analytically suppress singularities for both the two-body and three-body problems [\[320\]](#). However, Szebehely [\[319\]](#) shows that the use of a simple regularization technique, while analytically improving the problem, also results in an increased complexity of the equations of motion, which may degrade computational performance. In order to improve the numerical behavior of the equations of motion, more complicated regularization transformations have been introduced [\[36, 184, 324, 59, 186\]](#). In the present document, only the simple Sundman type transformations are considered, leaving other regularization techniques for future work. The three transformations considered are regularizations of close approaches to m_1 , m_2 , or both. These can be written as:

$$dt = s(\mathbf{r}, \mathbf{r}_{m_1}) d\tau = s d\tau \quad (2.48)$$

where $s = r_1 = \|\mathbf{r} - \mathbf{r}_{m_1}\|$, $s = r_2 = \|\mathbf{r} + c\mathbf{r}_{m_1}\|$ or $s = r_1 r_2 = \|\mathbf{r} - \mathbf{r}_{m_1}\| \|\mathbf{r} + c\mathbf{r}_{m_1}\|$, respectively. When applying the transformation to [Eq. \(2.35\)](#), the equations

of motion of the CRTBP become:

$$\begin{cases} \mathbf{X}' = \begin{bmatrix} s\mathbf{v} \\ s(p\mathbf{r} + q\mathbf{r}_{m_1}) \end{bmatrix} \\ t' = s \end{cases} \quad (2.49)$$

Note that this is the “time-velocity” version of the Sundman transformed inertial equations of motion.

Chapter 3

Multiple-Shooting Differential Dynamic Programming

Multiple-shooting has been demonstrated to benefit both direct and indirect methods for optimal control, but has yet to appear in the DDP literature. The present chapter¹ is aimed at applying the principles of multiple-shooting transcription to one flavor of DDP algorithms, adapted from Lantoine and Russell's HDDP algorithm. Associating multiple-shooting principles to the DDP framework has several benefits: first, as in all multiple-shooting methods, the sensitivity of the problem is reduced when splitting it into multiple subintervals, which improves the convergence properties and robustness of the algorithm; second, the inner loop on each multiple-shooting subinterval can be solved independently, and the formulation is therefore well adapted to parallel computing; finally, the use of DDP for the optimization of the controls partly alleviates the high computational complexity generally associated with

¹Work from this chapter was presented as:

- **Etienne Pellegrini** and Ryan P. Russell, A Multiple-Shooting Differential Dynamic Programming Algorithm, Paper AAS 17-453, *AAS/AIAA Space Flight Mechanics Meeting*, San Antonio, TX, February 2017.
- **Etienne Pellegrini** and Ryan P. Russell, Quasi-Newton Differential Dynamic Programming for Robust Low-Thrust Optimization, *AIAA/AAS Astrodynamics Specialists Conference*, Minneapolis, MN, August 2012, [doi:10.2514/6.2012-4425](https://doi.org/10.2514/6.2012-4425)

traditional direct multiple-shooting methods, and improves the overall robustness of the algorithm. The resulting algorithm is currently dubbed MDDP (Multiple-shooting Differential Dynamic Programming). First, the problem formulation is adapted to the multiple-shooting framework. The Augmented Lagrangian strategy utilized for the treatment of path and terminal constraints is explained. Then, the necessary quadratic expansions and update equations are developed for the single-leg problem, the multi-leg NLP solver, and the Lagrange multipliers update. These necessary theoretical developments augment the HDDP algorithm of Refs. [178] and [179], which constitute an important primer to the present chapter. The main differences between MDDP and HDDP are highlighted and explained. The structure of the MDDP algorithm and its main steps are presented. Finally, quasi-Newton methods are applied to the computation of the second-order STMs, in efforts to improve the computational efficiency of the algorithm.

3.1 The Multiple-Phase Optimal Control Problem Formulation

The MDDP algorithm is aimed at solving multiple-phase optimal control problems. The phases are defined by a change in any of the continuous functions \mathbf{f} , \mathbf{l} , \mathbf{g} (dynamics, cost, or constraints). In the case of interplanetary trajectory optimization, a change in the orbited body or in the thrusting engine for instance would correspond to a new phase. In all of the following developments, a superscript $[i]$ indicates that the scalar, vector, matrix or function is defined for the i^{th} phase. The general m -phase optimal control

problem in the Bolza form is expressed as the minimization with respect to the state, the controls, and the initial and final times, of the cost functional J defined as:

$$\begin{aligned} J &= \sum_{i=1}^m J^{[i]} \\ &= \sum_{i=1}^m \int_{t_0^{[i]}}^{t_f^{[i]}} l^{[i]}(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t), t) dt + \varphi^{[i]}(\mathbf{x}^{[i]}(t_0^{[i]}), t_0^{[i]}, \mathbf{x}^{[i]}(t_f^{[i]}), t_f^{[i]}) \end{aligned} \quad (3.1)$$

and subject to, for all $i \in [1, \dots, m]$:

- Dynamics equation:

$$\dot{\mathbf{x}}^{[i]}(t) = \mathbf{f}^{[i]}(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t), t) \quad (3.2)$$

- Path constraints:

$$\mathbf{g}^{[i]}(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t), t) \leq \mathbf{0} \quad (3.3)$$

- And finally, boundary and inter-phase linkage conditions:

$$\psi(\mathbf{x}^{[i]}(t_0^{[i]}), t_0^{[i]}, \mathbf{x}^{[i]}(t_f^{[i]}), t_f^{[i]} \mid i \in [1, \dots, m]) \leq \mathbf{0} \quad (3.4)$$

where, for phase i , $\mathbf{x}^{[i]} \in \mathbb{R}^{n_x}$ is the state vector, $\mathbf{u}^{[i]} \in \mathbb{R}^{n_u}$ is the controls vector, $\mathbf{f}^{[i]} : \mathbb{R}^{n_x+n_u+1} \rightarrow \mathbb{R}^{n_x}$ is the dynamics function, $l^{[i]} : \mathbb{R}^{n_x+n_u+1} \rightarrow \mathbb{R}$ is an accumulated cost, $\varphi^{[i]} : \mathbb{R}^{n_x+n_x+1} \rightarrow \mathbb{R}$ is the final cost, $\mathbf{g}^{[i]} : \mathbb{R}^{n_x+n_u+1} \rightarrow \mathbb{R}^{n_g}$ is the constraint function, and $\psi : \mathbb{R}^{2mn_x+2m} \rightarrow \mathbb{R}^{n_\psi}$ is the boundary function. Note that the boundary conditions vector function includes all phases, and is a function of all of their initial and final states. While the following developments can be adapted to having different dimensions for the state vector and control

vector (n_x and n_u respectively) for each phase, the current implementation holds these dimensions constant for all phases.

One of the notable differences between the standard optimal control problem formulation (including that of Ref. [178]) and the formulation used in this dissertation is the absence of static parameters (usually noted \mathbf{w} or \mathbf{p}). For conciseness purposes, they are included in the state vector, with their time derivative taken to be zero.

The first step in solving this continuous optimal control problem is to reformulate the very general Bolza form problem in a problem of Mayer. To do so, the state vector and dynamics function for each phase i are augmented as such:

$$\dot{\tilde{\mathbf{x}}}(t)^{[i]} = \begin{bmatrix} \mathbf{x}^{[i]}(t) \\ y^{[i]}(t) \end{bmatrix} = \tilde{\mathbf{f}}^{[i]}(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t), t) = \begin{bmatrix} \mathbf{f}^{[i]}(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t), t) \\ l^{[i]}(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t), t) \end{bmatrix} \quad (3.5)$$

Therefore, the accumulated cost $\int_{t_0^{[i]}}^{t_f^{[i]}} l^{[i]}(\mathbf{x}^{[i]}(t), \mathbf{u}^{[i]}(t), t) dt$ is now accounted for over the whole phase by adding $y^{[i]}(t_f^{[i]})$ to the terminal cost:

$$\hat{\varphi}^{[i]} = \varphi^{[i]} + y^{[i]}(t_f^{[i]}) \quad (3.6)$$

Then, the problem is discretized: the time interval $[t_0^{[i]}, \dots, t_f^{[i]}]$ is split into $n^{[i]}$ segments, corresponding to $n^{[i]} + 1$ nodes (see Fig. 3.1), with:

$$t_k^{[i]} = t_0^{[i]} + k \frac{t_f^{[i]} - t_0^{[i]}}{n^{[i]}}, \quad k = [0, \dots, n^{[i]}] \quad (3.7)$$

In the following, the k subscript denotes the value of a scalar, vector or matrix for the k^{th} segment. The control vector is constant over a segment. Note

that the controls for the last node of each phase are not used. The state is further augmented by adding the current time and the duration of the phase (denoted as time of flight or tof), which removes the explicit dependency of the dynamics, cost and controls on the independent variable:

$$\mathbf{X}_k^{[i]} = \begin{bmatrix} \tilde{\mathbf{x}}^{[i]}(t_k^{[i]}) \\ t_k^{[i]} \\ \text{tof}^{[i]} \end{bmatrix} \quad (3.8)$$

The new multiple-phase, discrete, constrained optimal control problem is expressed as follows. Minimize the cost:

$$J = \sum_{i=1}^m \hat{\varphi}^{[i]}(\mathbf{X}_0^{[i]}, \mathbf{X}_{n^{[i]}}^{[i]}) \quad (3.9)$$

subject to, for all $i \in [1, \dots, m]$ and all $k \in [0, \dots, n^{[i]}]$:

- Dynamics equation:

$$\mathbf{X}_{k+1}^{[i]} = \mathbf{F}^{[i]}(\mathbf{X}_k^{[i]}, \mathbf{u}_k^{[i]}) \quad (3.10)$$

- Path constraints:

$$\mathbf{g}^{[i]}(\mathbf{X}_k^{[i]}, \mathbf{u}_k^{[i]}) = \mathbf{g}_k^{[i]} \leq \mathbf{0} \quad (3.11)$$

- Boundary and linkage conditions

$$\boldsymbol{\psi}(\mathbf{X}_0^{[i]}, \mathbf{X}_{n^{[i]}}^{[i]}, \mid i \in [1, \dots, m]) \leq \mathbf{0} \quad (3.12)$$

Introducing the multiple-shooting legs.

The multiple-phase formulation of [Section 3.1](#) is adapted to a multiple-shooting solution to the problem: the problem is already discretized into

subintervals (the phases), with linkage conditions. However, in order to be able to control the number of multiple-shooting subintervals, an additional level of discretization is added: each phase is split into legs, which constitute the building blocks of the multiple-shooting algorithm. Each phase consists of $n_l^{[i]}$ legs, where $n_l^{[i]} \geq 1$ is selected by the user to control the sensitivity and ability to be parallelized. The total number of legs for the problem is $n_l = \sum_{i=1}^m n_l^{[i]}$. The superscript (j) is used for variables defined for the j^{th} leg, which is formed of $n^{(j)}$ segments (yielding $n^{[i]} = \sum_{j=1}^{n_l^{[i]}} n^{(j)}$). The different levels of discretization are depicted in Fig. 3.1. New intra-phase linkage conditions are introduced between legs. Within a phase, the succession of legs is by definition continuous, and a simple continuity condition is used:

$$\mathbf{X}_0^{(j+1)} = \begin{bmatrix} \mathbf{x}_0^{(j+1)} \\ y_0^{(j+1)} \\ t_0^{j+1} \\ \text{tof}^{[i]} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_{n^{(j)}}^{(j)} \\ 0 \\ t_{n^{(j)}}^{(j)} \\ \text{tof}^{[i]} \end{bmatrix} \quad (3.13)$$

Note that the integral cost y can be reinitialized at 0 at the beginning of each leg, and each internal leg has cost $\varphi^{(j)} = y_{n^{(j)}}^{(j)}$. The additivity property of the integral allows the removal of one continuity condition, a potential benefit to nonlinear problems. Equation (3.13) defines the intra-phase continuity condition. For the last leg of each phase, the general linkage conditions of Eq. (3.12) are used. Should an application necessitate the continuity of higher-order derivatives of the motion between legs, users have two different possibilities. Adding the higher-order derivatives to the state will immediately ensure their continuity through Eq. (3.13). However, this will also increase the dimension of the state and could be detrimental to the computational efficiency of the

algorithm. The user can also add the higher-order derivatives continuity to the general linkage conditions of Eq. (3.12), and multiply the number of phases instead of adding legs.

In order to solve the problem using multiple-shooting principles, the vector \mathbf{s} of NLP variables is introduced, and is formed with the initial states of each of the multiple-shooting subintervals or legs:

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}^{(1)} \\ \mathbf{s}^{(2)} \\ \dots \\ \mathbf{s}^{(n_l)} \end{bmatrix} = \begin{bmatrix} \mathbf{X}_0^{(1)} \\ \mathbf{X}_0^{(2)} \\ \dots \\ \mathbf{X}_0^{(n_l)} \end{bmatrix} \quad (3.14)$$

Finally, the optimal control problem solved in the remainder of the chapter is the following: minimize the cost functional J defined as:

$$J = \sum_{j=1}^{n_l} \hat{\varphi}^{(j)}(\mathbf{X}_0^{(j)}, \mathbf{X}_{n^{(j)}}^{(j)}) = \sum_{j=1}^{n_l} \hat{\varphi}^{(j)}(\mathbf{s}^{(j)}, \mathbf{X}_{n^{(j)}}^{(j)}) \quad (3.15)$$

subject to, for all $j \in [1, \dots, n_l]$ and all $k \in [0, \dots, n^{(j)}]$:

- Dynamics equation:

$$\mathbf{X}_{k+1}^{(j)} = \mathbf{F}^{(j)}(\mathbf{X}_k^{(j)}, \mathbf{u}_k^{(j)}) \quad (3.16)$$

- Path constraints:

$$\mathbf{g}^{(j)}(\mathbf{X}_k^{(j)}, \mathbf{u}_k^{(j)}) = \mathbf{g}_k^{(j)} \leq \mathbf{0} \quad (3.17)$$

- Boundary and linkage conditions

$$\tilde{\psi}(\mathbf{s}, \mathbf{X}_{n^{(j)}}^{(j)}, \mid j \in [1, \dots, n_l]) \leq \mathbf{0} \quad (3.18)$$

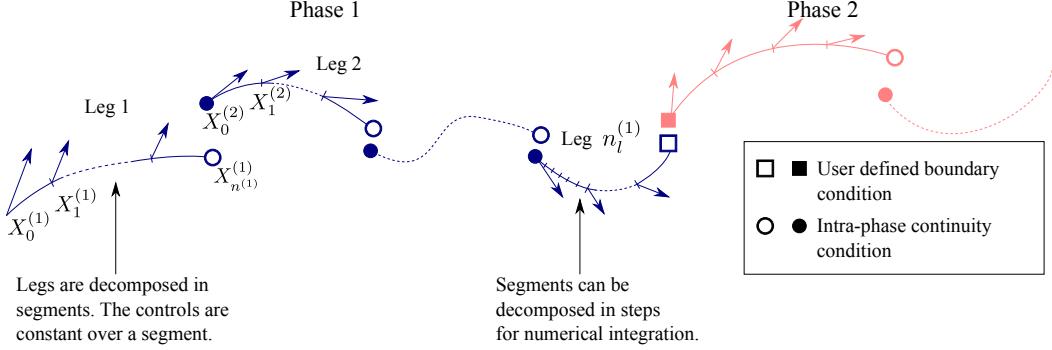


Figure 3.1: Discretization of the optimal control problem for two phases

where: $\hat{\varphi}^{(j)} = \hat{\varphi}^{[i]}$ if the j^{th} leg is the final leg of a phase, $\hat{\varphi}^{(j)} = y_{n^{(j)}}^{(j)}$ if it is an inner leg. \mathbf{F} and \mathbf{g} are, by definition, identical over all the legs of a phase, and $\tilde{\psi}$ contains the inter-phase boundary conditions of Eq. (3.12) for the final legs of phases, augmented with the intra-phase continuity conditions of Eq. (3.13) for the inner legs. Therefore, from an algorithmic point of view, once the dynamics, cost and constraints functions have been correctly defined, the phases are no longer relevant to the formulation, and the process moves forward in the context of legs only.

The goal of the work presented in this chapter is to solve the problem above using multiple-shooting and DDP principles: a variant of the HDDP algorithm is used to solve for the controls on each leg, and form the necessary sensitivities for a general NLP algorithm, dubbed “multi-leg NLP solver”, to solve for the multiple-shooting variable \mathbf{s} .

The problem to be solved by DDP, dubbed the “single-leg problem”, is to minimize the cost functional $J^{(j)} = \hat{\varphi}^{(j)}(\mathbf{X}_0^{(j)}, \mathbf{X}_{n^{(j)}}^{(j)})$ with respect to $\mathbf{u}_0^{(j)}$, $\mathbf{u}_1^{(j)}, \dots, \mathbf{u}_{n^{(j)}-1}^{(j)}$ subject to, for all $k \in [0, \dots, n^{(j)}]$:

- Dynamics equation:

$$\mathbf{X}_{k+1}^{(j)} = \mathbf{F}^{(j)}(\mathbf{X}_k^{(j)}, \mathbf{u}_k^{(j)}) \quad (3.19)$$

- Path constraints:

$$\mathbf{g}^{(j)}(\mathbf{X}_k^{(j)}, \mathbf{u}_k^{(j)}) = \mathbf{g}_k^{(j)} \leq \mathbf{0} \quad (3.20)$$

- Boundary and linkage conditions

$$\psi^{(j)}(\mathbf{X}_{n^{(j)}}^{(j)}, \mathbf{s}) \leq \mathbf{0} \quad (3.21)$$

where $\psi^{(j)}$ is a vector containing all the elements of $\tilde{\psi}$ that depend on $\mathbf{X}_{n^{(j)}}^{(j)}$, and is of dimension $n_\lambda^{(j)}$.

3.2 Augmented Lagrangian Algorithm for Constrained Optimization

The MDDP algorithm is aimed at solving constrained optimal control problems. The treatment of the general path and terminal constraints $\mathbf{g}^{(j)}$ and $\psi^{(j)}$ introduced in Eq. (3.20) and Eq. (3.21), respectively, is a crucial part of the algorithm. An Augmented Lagrangian approach is chosen to handle all constraints, with the exception of simple bounds on the controls [Eq. (3.22)] and on the initial states of each leg [Eq. (3.23)] which, when present, are extracted from $\mathbf{g}^{(j)}$ and $\psi^{(j)}$ and handled using a null-space trust-region method, as described in Section 3.4.1.

$$\mathbf{u}_l^{(j)} \leq \mathbf{u}_k^{(j)} \leq \mathbf{u}_u^{(j)}, \quad \forall k \in [0, \dots, n^{(j)}] \quad (3.22)$$

$$\mathbf{X}_l^{(j)} \leq \mathbf{X}_0^{(j)} \leq \mathbf{X}_u^{(j)} \quad (3.23)$$

The Augmented Lagrangian approach is a common method of solving constrained problems with unconstrained optimization techniques, which has been demonstrated to be less susceptible to ill-conditioning than penalty and barrier methods (see [Section 1.2.1.3](#)). It is particularly adapted to the MDDP framework, since it allows to treat constraints while keeping the unconstrained trust-region algorithm as the main component for the optimization of the quadratic subproblems (see [Section 3.3](#) and [Section 3.4.1](#)). Moreover, the structure of the MDDP algorithm, and in particular the use of the first- and second-order STMs for sensitivity propagation, permits the computation of second-order partials of the cost function with respect to the Lagrange multipliers for a low computational cost, making the use of a second-order formula to update some of the multipliers more efficient and practical.

3.2.1 Equality Constrained Problems

The main principles of the Augmented Lagrangian approach for equality constraints are explained in this section. A more detailed analysis, as well as proofs of principles and of convergence can be found in a number of textbooks, including the comprehensive book by Bertsekas [\[22\]](#). The problem to be solved in this section is presented in [Eq. \(3.24\)](#):

$$\underset{\mathbf{X}}{\text{minimize}} \quad J = \varphi(\mathbf{X}) \quad \text{subject to} \quad \psi(\mathbf{X}) = 0 \quad (3.24)$$

The main idea behind Augmented Lagrangian methods is to modify the problem to include a penalty term, which in this work is chosen to be quadratic:

$$\underset{\mathbf{X}}{\text{minimize}} \quad \hat{J} = \varphi(\mathbf{X}) + c\|\psi(\mathbf{X})\|^2 \quad \text{subject to} \quad \psi(\mathbf{X}) = 0 \quad (3.25)$$

The penalty term does not change the solution of the original problem, but helps ensuring convexity of the Hessian of the Lagrangian, while using the Lagrangian prevents the ill-conditioning present in pure penalty methods. The resulting augmented cost is:

$$\tilde{J}(\mathbf{X}, \boldsymbol{\lambda}, c) = \varphi(\mathbf{X}) + \boldsymbol{\lambda}^T \boldsymbol{\psi}(\mathbf{X}) + c \|\boldsymbol{\psi}(\mathbf{X})\|^2 \quad (3.26)$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers, and $c > 0$ is called the penalty parameter. The primal-dual philosophy of original Lagrangian methods [22, 9] is conserved: the optimal parameters and Lagrange multipliers can be obtained by maximizing the dual functional $d_c(\boldsymbol{\lambda})$ defined in Eq. (3.27). The constrained optimization problem is replaced with two unconstrained subproblems:

$$\underset{\boldsymbol{\lambda}}{\text{maximize}} \quad d_c(\boldsymbol{\lambda}) \quad \text{with} \quad d_c(\boldsymbol{\lambda}) = \min_{\mathbf{X}} \tilde{J}(\mathbf{X}, \boldsymbol{\lambda}, c) \quad (3.27)$$

It can be demonstrated that there exist a value $c^* < +\infty$ such that for, $c \geq c^*$, and provided sufficient assumptions to interchange limits and infima, Problem (3.24) and Problem (3.27) have the same solution [22]. In Augmented Lagrangian algorithms, the classic quadratic penalty method is essentially combined with primal-dual Lagrangian methods, which improves the convergence properties of the algorithm. In the case of typical penalty methods, the penalty parameter c has to be increased to infinity to guarantee the respect of the constraints, creating an obvious ill-conditioning problem. The addition of the Lagrange multiplier allows for the problem to formally converge for lower values of c (i.e. less than infinity). Moreover, usual primal-dual Lagrangian algorithms can suffer from unboundedness when the Hessian of the Lagrangian

is non positive-definite, a problem that is remedied by the penalty term, once $c > c^*$.

The solution method for [Problem \(3.27\)](#) employs a two-loop philosophy: an inner loop, which can be based on any unconstrained optimization method, is responsible for solving for \mathbf{X} and effectively evaluates the dual functional. In original Augmented Lagrangian algorithms, the algorithm only proceeds to the outer loop once the inner loop is converged [260], whereas modern algorithms utilize the notion of approximate convergence [75], which relaxes the inner-loop convergence tolerances by orders of magnitude. The approximate convergence strategy is adopted in MDDP, as explained in [Section 3.4.3](#). Note that the convergence criteria at this point does not include the constraints, since the inner loop solves an unconstrained optimization problem. Once convergence or approximate convergence has been reached, an outer loop is responsible for the penalty parameter and Lagrange multipliers update. If the value c^* of the minimum penalty parameter for convergence could be predicted, any $c > c^*$ could be used from the first iteration of the algorithm. However, c^* being unknown forces an update strategy, as picking arbitrary values for c could result in ill-conditioning, or non-convexity of the augmented problem. The penalty parameter usually follows a predetermined strictly increasing sequence and is increased whenever the constraints are grossly non-satisfied during an outer loop iteration. A larger penalty parameter will force the solution of the next inner loop solve towards feasibility. When the constraints are approximately respected, the Lagrange multipliers can be updated, to move towards feasibility and optimality. A number of different update methods for the Lagrange

multipliers have been proposed in the literature [75, 150, 321], the most popular being a simple first-order update:

$$\delta \boldsymbol{\lambda} = 2c\boldsymbol{\psi}(\mathbf{X}) \quad (3.28)$$

This update is in the steepest ascent direction, since $\nabla_{\boldsymbol{\lambda}} \tilde{J}(\mathbf{X}, \boldsymbol{\lambda}) = \boldsymbol{\psi}(\mathbf{X})$. The magnitude of the update comes from the fact that, at iteration p of the outer loop of an Augmented Lagrangian algorithm, the gradient of the augmented cost is:

$$\frac{\partial \tilde{J}}{\partial \mathbf{X}} \Bigg|_{(\mathbf{X}_p, \boldsymbol{\lambda}_p, c_p)} = \varphi_X(\mathbf{X}_p) + [\boldsymbol{\lambda}_p + 2c_p \boldsymbol{\psi}(\mathbf{X}_p)] \boldsymbol{\psi}_X(\mathbf{X}_p) \quad (3.29)$$

The sequence $(\mathbf{X}_p, \boldsymbol{\lambda}_p)$ converges towards the optimal point $(\mathbf{X}^*, \boldsymbol{\lambda}^*)$, where the Karush-Kuhn-Tucker (KKT) conditions are respected:

$$\varphi_X(\mathbf{X}^*) + \boldsymbol{\lambda}^* \boldsymbol{\psi}_X(\mathbf{X}^*) = 0 \quad (3.30)$$

$$\varphi_X(\mathbf{X}^*) = -\boldsymbol{\lambda}^* \boldsymbol{\psi}_X(\mathbf{X}^*) \quad (3.31)$$

Comparison of Eq. (3.29) and Eq. (3.31) suggests that the sequence $\boldsymbol{\lambda}_{p+1} = \boldsymbol{\lambda}_p + 2c_p \boldsymbol{\psi}(\mathbf{X}_p)$ converges towards $\boldsymbol{\lambda}^*$, and justifies the magnitude of the first-order update. In the MDDP algorithm, this first-order update is applied to the path constraints multipliers, while the terminal constraints multipliers employ a second-order update, as explained in Section 3.2.3.

3.2.2 Extension to Inequality Constraints

The principles of the Augmented Lagrangian method for equality constraints have been introduced in Section 3.2.1. However, most optimal control problems involve a number of inequality constraints, the treatment of which

is presented in this section. A common way to handle inequality constraints in optimization is to transform them into equality constraints by introducing slack variables [155, pp. 136-138], henceforth denoted $\boldsymbol{\alpha}$. The general inequality constrained problem is described in Eq. (3.32):

$$\underset{\mathbf{X}}{\text{minimize}} \quad J = \varphi(\mathbf{X}) \quad \text{subject to} \quad \psi(\mathbf{X}) \leq 0 \quad (3.32)$$

Adding slack variables allows to rewrite the problem as:

$$\underset{\mathbf{X}, \boldsymbol{\alpha}}{\text{minimize}} \quad J = \varphi(\mathbf{X}) \quad \text{subject to} \quad \hat{\psi}(\mathbf{X}, \boldsymbol{\alpha}) \equiv \psi(\mathbf{X}) + \boldsymbol{\alpha} = 0, \quad \boldsymbol{\alpha} \geq 0 \quad (3.33)$$

or alternatively:

$$\underset{\mathbf{X}, \boldsymbol{\alpha}}{\text{minimize}} \quad J = \varphi(\mathbf{X}) \quad \text{subject to} \quad \hat{\psi}(\mathbf{X}, \boldsymbol{\alpha}) \equiv \psi(\mathbf{X}) + \boldsymbol{\alpha}^2 = 0 \quad (3.34)$$

where $\boldsymbol{\alpha}^2(i) = \alpha(i)^2$. The new problem, which contains only equality constraints, can be solved using the strategy described in Section 3.2.1, where the inner loop now solves for the states \mathbf{X} as well as $\boldsymbol{\alpha}$. An obvious problem of this strategy is the addition of n_ψ new parameters, where n_ψ is the number of inequality constraints, increasing the complexity of the inner loop problem.

Rockafellar [283] introduces a way to eliminate the slack variables from the optimization of the inner loop, by noticing the convexity of the augmented cost of Problem (3.33) with respect to the individual slack variables. The minimization with respect to each α_i can be carried out explicitly. The augmented cost is written:

$$\tilde{J}(\mathbf{X}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, c) = \varphi(\mathbf{X}) + \boldsymbol{\lambda}^T[\psi(\mathbf{X}) + \boldsymbol{\alpha}] + c\|\psi(\mathbf{X}) + \boldsymbol{\alpha}\|^2 \quad (3.35)$$

There is no cross-terms between \mathbf{X} and $\boldsymbol{\alpha}$, and the new minimization problem in $\boldsymbol{\alpha}$ is written:

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \quad \tilde{J}(\mathbf{X}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, c) = \boldsymbol{\lambda}^T \boldsymbol{\alpha} + c\|\boldsymbol{\psi}(\mathbf{X}) + \boldsymbol{\alpha}\|^2 \quad \text{subject to} \quad \boldsymbol{\alpha} \geq 0 \quad (3.36)$$

The unconstrained minimizer is obtained when the first-order partial is canceled:

$$\tilde{J}_\alpha(\mathbf{X}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, c) = \boldsymbol{\lambda}^T + 2c(\boldsymbol{\psi}(\mathbf{X}) + \boldsymbol{\alpha}^{*,\text{unc}})^T = 0 \quad (3.37)$$

$$\boldsymbol{\alpha}^{*,\text{unc}} = -\frac{\boldsymbol{\lambda}}{2c} - \boldsymbol{\psi}(\mathbf{X}) \quad (3.38)$$

The problem is convex in $\boldsymbol{\alpha}$ since $\tilde{J}_{\alpha\alpha}(\mathbf{X}, \boldsymbol{\lambda}, \boldsymbol{\alpha}, c) = 2c \mathbb{I}_{n_\psi \times n_\psi} \succ 0$. Therefore, the unconstrained minimizer is a global minimizer, and, for any $i \in [1, \dots, n_\psi]$, if $\alpha_i^{*,\text{unc}} < 0$ then $\alpha_i^* = 0$. Finally, the solution of the inner loop problem with respect to α_i is:

$$\alpha_i^* = \begin{cases} -\frac{\lambda_i}{2c} - \psi_i(\mathbf{X}) & \text{if } \psi_i(\mathbf{X}) \leq -\frac{\lambda_i}{2c} \\ 0 & \text{otherwise} \end{cases} \quad (3.39)$$

As in typical inequality constraints treatment, the i^{th} constraint is considered active when $\alpha_i = 0$, and inactive otherwise. Note that this solution essentially shifts the boundary value for a constraint to be activated or deactivated from $\psi_i(\mathbf{X}) = 0$ to $\psi_i(\mathbf{X}) = -\frac{\lambda_i}{2c}$. [Equation \(3.39\)](#) can now be plugged into the augmented cost of [Eq. \(3.35\)](#), which yields the new slack-free augmented cost:

$$\tilde{J}(\mathbf{X}, \boldsymbol{\lambda}, c) = \varphi(\mathbf{X}) + \sum_{i=1}^{n_\alpha} P(\psi_i(\mathbf{X}), \lambda_i, c) \quad (3.40)$$

with

$$P(\psi_i(\mathbf{X}), \lambda_i, c) = \begin{cases} -\frac{\lambda_i^2}{4c} & \text{if } \psi_i(\mathbf{X}) \leq -\frac{\lambda_i}{2c} \\ \lambda_i \psi_i(\mathbf{X}) + c \psi_i(\mathbf{X})^2 & \text{otherwise} \end{cases} \quad (3.41)$$

The Augmented Lagrangian approach of [Section 3.2.1](#) can now be used to solve the primal-dual problem based on the slack-free augmented cost of [Eq. \(3.40\)](#). The KKT conditions state that the Lagrange multipliers associated with inequality constraints are nonnegative. The multipliers update is therefore modified to enforce this nonnegativity:

$$\lambda_{i,\text{new}} = \max(\lambda_i + \delta\lambda_i, 0) \quad (3.42)$$

where $\delta\lambda_i$ can be computed as in [Eq. \(3.28\)](#), or using more complex updates, such as the second-order update described [Section 3.3.3](#).

It is important to note that the slack-free augmented cost of [Eq. \(3.40\)](#) is continuously differentiable with respect to \mathbf{X} , but presents a discontinuity in the second-order partials. Most authors quickly dismiss the discontinuity, since after a sufficient number of iterations p , the minimizer \mathbf{X}_p of the inner loop is usually far from the nonsmooth regions [283, 230]. In a DDP algorithm based on quadratic expansions and on the computation of an expected reduction, such as in MDDP, the discontinuity can create difficulties, in particular at earlier iterations where constraints can go through the active/inactive boundary regularly. In order to address this problem, several strategies have been implemented and studied, including using a Heaviside approximation or a polynomial interpolation to smooth the transition. However, no fully satisfactory solution has been found yet, and the smoothing of the slack-free augmented cost is left for future work. The implementation of MDDP presented in this dissertation does not smooth the constraints, and can suffer from reduced trust-region radii when going through the second-order discontinuity.

The iteration counts presented in [Chapter 4](#) can likely be improved with an adapted smoothing strategy.

3.2.3 Application to the MDDP algorithm

In order to apply the principles explained in [Section 3.2.1](#) and [Section 3.2.2](#) to the MDDP algorithm, it is important to first define the type of constraints present in the single-leg problem from [Section 3.1](#). For leg (j) , there are $n_\lambda^{(j)}$ terminal constraints, and $n_g^{(j)}$ path constraints. However, the path constraints have to be satisfied at each segment, effectively creating $n^{(j)} \times n_g^{(j)}$ constraints and associated Lagrange multipliers. The terminal constraints depend on the final state of the leg, $\mathbf{X}_n^{(j)}$, and the vector of multiple-shooting variables \mathbf{s} . The path constraints at segment k depend on the state $\mathbf{X}_k^{(j)}$ and the controls $\mathbf{u}_k^{(j)}$, for all $k \in [0, \dots, n^{(j)}]$. In the following, the Lagrange multipliers and penalty parameter associated with the terminal constraints are noted $\boldsymbol{\lambda}^{(j)}$ and $c^{(j)}$, respectively, while those associated with the path constraints are noted $\boldsymbol{\mu}^{(j)}$ and $d^{(j)}$, respectively. For each segment k , the vector $\boldsymbol{\mu}_k^{(j)}$ is of size $n_g^{(j)} \times 1$, since the path constraints are to be satisfied at every segment. The total (slack-free) augmented cost for leg (j) is written:

$$\begin{aligned} J^{(j)} = & \sum_{k=0}^{n^{(j)}} L(\mathbf{g}_k^{(j)}(\mathbf{X}_k^{(j)}, \mathbf{u}_k^{(j)}), \boldsymbol{\mu}_k^{(j)}, d^{(j)}) \\ & + \hat{\varphi}^{(j)}(\mathbf{X}_0^{(j)}, \mathbf{X}_n^{(j)}) + \sum_{i=0}^{n_\psi^{(j)}} P(\psi_i^{(j)}(\mathbf{X}_n^{(j)}, \mathbf{s}), \lambda_i^{(j)}, c^{(j)}) \end{aligned} \quad (3.43)$$

with

$$L(\mathbf{g}, \boldsymbol{\mu}, d) = \sum_{i=0}^{n_g^{(j)}} P(g_i, \mu_i, d) \quad (3.44)$$

and

$$P(h, \theta, r) = \begin{cases} -\frac{\theta^2}{4r} & \text{if } h \leq -\frac{\theta}{2r} \\ \theta h + rh^2 & \text{otherwise} \end{cases} \quad (3.45)$$

The Augmented Lagrangian framework described in the previous two sections can be applied to this augmented cost. The path constraints augmentation is equivalent to a running cost, and is treated in a similar fashion to the running cost of Ref. [178]. Because of the potentially high number of path constraints Lagrange multipliers, the vectors $\lambda^{(j)}$ and $\mu_k^{(j)}$, $k \in [0, \dots, n^{(j)}]$ are treated differently: as detailed in Section 3.3.3, the terminal constraints multipliers are updated with a second-order formula, and have a feedback effect on the control law. A similar treatment of the path constraints could result in the need to optimize very large problems, inverting matrices with thousands to tens of thousands of elements, and therefore partly negating the advantages of using a DDP method. For this reason, the simple (uncoupled) first-order update introduced in Eq. (3.28) and Eq. (3.42) is used for μ_k :

$$\mu_{i,k,\text{new}}^{(j)} = \max\left([\mu_{i,k}^{(j)} + 2d^{(j)}g^{(j)}(\mathbf{X}_k^{(j)})], 0\right) \quad (3.46)$$

The update is applied when the inner loop is approximately converged, and the path constraints approximately satisfied (see Section 3.4 and Section 3.4.3). Moreover, the current implementation does not include feedback terms in μ in the control law. The path constraints are essentially treated independently from the other parameters of the problem, allowing for their simple and efficient treatment even with high dimensions.

In all of the developments above, the penalty parameters c and d are monotonically increasing throughout the outer loop iterations, until the termi-

nal constraints and path constraints are approximately satisfied, as described in [Section 3.4](#). A number of different update formulas are available in the literature, but a simple predetermined sequence is used in the present implementation:

$$c_{\text{new}}^{(j)} = \gamma_c c^{(j)} \quad (3.47)$$

$$d_{\text{new}}^{(j)} = \gamma_d d^{(j)} \quad (3.48)$$

Finally, it should be noted that each major iteration of the Augmented Lagrangian algorithm is associated with an update of the tolerances used to determinate approximate convergence. The idea is described by Conn et al. [[75](#)], who use it to prove the convergence of their algorithm, and is implemented in the optimization software LANCELOT [[77](#)]. Both the first-order optimality tolerance and the constraint satisfaction threshold are reduced with each iteration of the outer loop, forcing the subsequent solutions of the inner loop to be increasingly accurate. This process is similar to continuation methods, since an easier problem with relaxed constraints is solved first and used as an initial guess for the next iteration. The scheme used in MDDP is described in [Algorithm 1](#).

3.3 Solving the Multiple-Shooting Problem

The MDDP algorithm is designed according to the Augmented Lagrangian principles described in [Section 3.2](#), and, in particular, following the primal-dual philosophy. The inner loop of the complete multiple-shooting DDP algorithm consists in solving the one-leg problem for each of the legs, as well

as optimizing the vector \mathbf{s} of initial conditions of the legs. The outer loop is responsible for the update of the Lagrange multipliers and penalty parameters. The present section describes the use of DDP methods to solve the inner loop problem, as well as to compute the terminal constraints Lagrange multiplier update. The structure of the algorithm, with details on the organization of the inner and outer loops, is given in [Section 3.4](#).

3.3.1 Single-leg Quadratic Expansions

The elementary building block of the MDDP algorithm is the leg, or multiple-shooting subinterval. In order to solve for the optimal controls for each leg, as well as to obtain the partial derivatives necessary to solve the initial conditions NLP and update the Lagrange multipliers, a modified version of the HDDP algorithm developed by Lantoine and Russell is used. The structure of the solution method is the same. It is summarized in [Section 3.4](#), and a detailed description can be found in Ref. [178]. In the present section, focus is given to the quadratic expansions and update equations, which have to be adapted to the newly introduced dependence of the boundary conditions (and therefore of the augmented cost) on the multiple-shooting vector \mathbf{s} . In addition, as mentioned in [Section 3.1](#), the static parameters are included in the state and do not appear in the equations explicitly anymore. The running cost L is the augmentation due to the path constraints, as explained in [Section 3.2.3](#). In this section, since only one leg is considered at a time, the superscript (j) is dropped. The whole multiple-shooting vector is still designated with \mathbf{s} , while $\mathbf{s}^{(j)}$ is noted \mathbf{X}_0 .

First, the augmented cost-to-go function at segment n is formed. The cost-to-go is defined as the cost incurred from the current point until the end of the leg:

$$J_n = L(\mathbf{g}_n(\mathbf{X}_n, \mathbf{u}_n), \boldsymbol{\mu}_n, d) + \hat{\varphi}(\mathbf{X}_0, \mathbf{X}_n) + \sum_{i=0}^{n_\psi} P(\psi_i(\mathbf{X}, \mathbf{s}), \lambda_i, c) \quad (3.49)$$

with L and P as defined in Eq. (3.44) and Eq. (3.45) respectively. Therefore, the cost-to-go now has a dependency on the whole \mathbf{s} , vector through the boundary conditions, as well as a dependency on $\mathbf{X}_0 = \mathbf{s}^{(j)}$ through the original cost function $\hat{\varphi}$, in addition to the usual dependency on the states and controls. It is therefore necessary to develop quadratic expansions with respect to \mathbf{X} , \mathbf{u} , to obtain an optimal control law. The expansions with respect to \mathbf{s} are required to be able to solve the initial conditions NLP, and complete the solution of the inner loop problem. Finally, the expansions with respect to $\boldsymbol{\lambda}$ are used in the outer loop, to compute a second-order update of the terminal constraints Lagrange multipliers. The cost also has a dependency on the vector of path constraints $\boldsymbol{\mu}_n$. However, this dependency is ignored in the quadratic expansions, since the simple first-order update of Eq. (3.46) is used, and no feedback law in $\boldsymbol{\mu}$ is created. Expanding the cost-to-go function J_k with respect to \mathbf{X} ,

\mathbf{u} , $\boldsymbol{\lambda}$, and \mathbf{s} gives:

$$\begin{aligned}
\delta J_k = & ER_{k+1} + J_{X,k}^T \delta \mathbf{X}_k + J_{u,k}^T \delta \mathbf{u}_k + J_{\lambda,k}^T \delta \boldsymbol{\lambda} + J_{s,k}^T \delta \mathbf{s} \\
& + \frac{1}{2} \delta \mathbf{X}_k^T J_{XX,k} \delta \mathbf{X}_k + \frac{1}{2} \delta \mathbf{u}_k^T J_{uu,k} \delta \mathbf{u}_k + \frac{1}{2} \delta \boldsymbol{\lambda}^T J_{\lambda\lambda,k} \delta \boldsymbol{\lambda} + \frac{1}{2} \delta \mathbf{s}^T J_{ss,k} \delta \mathbf{s} \\
& + \delta \mathbf{X}_k^T J_{Xu,k} \delta \mathbf{u}_k + \delta \mathbf{X}_k^T J_{X\lambda,k} \delta \boldsymbol{\lambda} + \delta \mathbf{X}_k^T J_{Xs,k} \delta \mathbf{s} \\
& + \delta \mathbf{u}_k^T J_{u\lambda,k} \delta \boldsymbol{\lambda} + \delta \mathbf{u}_k^T J_{us,k} \delta \mathbf{s} \\
& + \delta \boldsymbol{\lambda}^T J_{\lambda s,k} \delta \mathbf{s}
\end{aligned} \tag{3.50}$$

The State-Transition Matrices (STMs) are used in order to obtain the partials necessary for the computation of Eq. (3.50). The upstream segments having already been optimized, the partials $J_{X,k+1}^*$, $J_{\lambda,k+1}^*$, $J_{s,k+1}^*$, $J_{XX,k+1}^*$, $J_{\lambda\lambda,k+1}^*$, $J_{ss,k+1}^*$, $J_{X\lambda,k+1}^*$, $J_{Xs,k+1}^*$, and $J_{\lambda s,k+1}^*$, are already known. The mapping of the optimized partials¹ at segment $k+1$ to the partials at segment k is as follows:

$$\begin{bmatrix} J_{X,k} \\ J_{u,k} \end{bmatrix}^T = \begin{bmatrix} L_{X,k} \\ L_{u,k} \end{bmatrix}^T + \begin{bmatrix} J_{X,k+1}^* \\ \mathbf{0}_{n_u} \end{bmatrix}^T \Phi_k^1 \tag{3.51}$$

$$\begin{bmatrix} J_{XX,k} & J_{Xu,k} \\ J_{uX,k} & J_{uu,k} \end{bmatrix} = \begin{bmatrix} L_{XX,k} & L_{Xu,k} \\ L_{uX,k} & L_{uu,k} \end{bmatrix} + \Phi_k^{1T} \begin{bmatrix} J_{XX,k+1}^* & \mathbf{0}_{n_X \times n_u} \\ \mathbf{0}_{n_u \times n_X} & \mathbf{0}_{n_u \times n_u} \end{bmatrix} \Phi_k^1 + \begin{bmatrix} J_{X,k+1}^* \\ \mathbf{0} \end{bmatrix}^T \bullet_1 \Phi_k^2 \tag{3.52}$$

where $\Phi_k^1 = \Phi^1(t_{k+1}, t_k)$ is the first-order STM (an $(n_X + n_u) \times (n_X + n_u)$ matrix), $\Phi_k^2 = \Phi^2(t_{k+1}, t_k)$ is the second-order STM (an $(n_X + n_u) \times (n_X + n_u) \times (n_X + n_u)$ tensor), and with the bullet operator defined in Section 2.1. Since neither the multipliers nor the \mathbf{s} vector appear in the equations of motion, the

¹Optimized partials are the partial derivatives that result from a backward sweep and apply when downstream control laws govern all control decisions.

mapping of their derivatives is straightforward:

$$J_{\lambda,k} = J_{\lambda,k+1}^* \quad (3.53)$$

$$J_{\lambda\lambda,k} = J_{\lambda\lambda,k+1}^* \quad (3.54)$$

$$J_{s,k} = J_{s,k+1}^* \quad (3.55)$$

$$J_{ss,k} = J_{ss,k+1}^* \quad (3.56)$$

$$J_{\lambda s,k} = J_{\lambda s,k+1}^* \quad (3.57)$$

Cross derivatives with the state use the first-order STM:

$$\begin{bmatrix} J_{X\lambda,k}^T & J_{u\lambda,k}^T \end{bmatrix} = \begin{bmatrix} J_{X\lambda,k+1}^{*T} & 0_{n_\lambda \times n_u} \end{bmatrix} \Phi_k^1 \quad (3.58)$$

$$\begin{bmatrix} J_{Xs,k}^T & J_{us,k}^T \end{bmatrix} = \begin{bmatrix} J_{Xs,k+1}^{*T} & 0_{n_s \times n_u} \end{bmatrix} \Phi_k^1 \quad (3.59)$$

The initialization of the sensitivities $J_{X,n}^*$, $J_{\lambda,n}^*$, $J_{s,n}^*$, $J_{XX,n}^*$, $J_{\lambda\lambda,n}^*$, $J_{ss,n}^*$, $J_{X\lambda,n}^*$, $J_{Xs,n}^*$, and $J_{\lambda s,n}^*$ is done by taking the partial derivatives of the cost-to-go at segment n , J_n .

All the terms in Eq. (3.50) are now well defined, and a control law can be obtained by minimizing the cost-to-go function with respect to $\delta \mathbf{u}_k$:

$$\delta \mathbf{u}_k = -J_{uu,k}^{-1} (J_{u,k} + J_{uX,k} \delta \mathbf{X}_k + J_{u\lambda,k} \delta \boldsymbol{\lambda} + J_{us} \delta \mathbf{s}) \quad (3.60)$$

Note that this control law has an extra term when compared to the one presented by Lantoine and Russell [178]: an additional feedback matrix is necessary to account for changes in the initial conditions generated by the NLP solver. The expression of Eq. (3.60) might generate controls that violate the bounds of Eq. (3.22). Moreover, if $J_{uu,k}$ is not positive-definite, the control

update has no guarantee to be a descent direction. In order to ensure positive-definiteness, respect of the control bounds to first order, and to limit the size of the step taken (so that the quadratic approximations stay valid), a null-space trust-region method is used (see [Section 3.4.1](#)). The trust-region method returns a shifted Hessian matrix $\tilde{J}_{uu,k}$, which is used in the new control law:

$$\delta \mathbf{u}_k = -\tilde{J}_{uu,k}^{-1}(J_{u,k} + J_{uX,k}\delta \mathbf{X}_k + J_{u\lambda,k}\delta \boldsymbol{\lambda} + J_{us}\delta \mathbf{s}) \quad (3.61)$$

$$\delta \mathbf{u}_k = A_k + B_k\delta \mathbf{X}_k + C_k\delta \boldsymbol{\lambda} + D_k\delta \mathbf{s} \quad (3.62)$$

with

$$A_k = -\tilde{J}_{uu,k}^{-1}J_{u,k} \quad (3.63)$$

$$B_k = -\tilde{J}_{uu,k}^{-1}J_{uX,k} \quad (3.64)$$

$$C_k = -\tilde{J}_{uu,k}^{-1}J_{u\lambda,k} \quad (3.65)$$

$$D_k = -\tilde{J}_{uu,k}^{-1}J_{us,k} \quad (3.66)$$

Note that D_k is the new control law resulting from the multiple-shooting formulation, providing feedback to individual legs due to changes in initial conditions of other legs. Replacing $\delta \mathbf{u}_k$ by the control law of [Eq. \(3.62\)](#) in [Eq. \(3.50\)](#) yields the following update equations for the expected reduction and the op-

timized partials:

$$ER_k = ER_{k+1} + J_{u,k}^T A_k + \frac{1}{2} A_k^T J_{uu,k} A_k \quad (3.67)$$

$$J_{X,k}^* = J_{X,k} + A_k^T J_{uu,k} B_k + J_{uX,k}^T A_k + B_k^T J_{u,k} \quad (3.68)$$

$$J_{\lambda,k}^* = J_{\lambda,k} + A_k^T J_{uu,k} C_k + J_{u\lambda,k}^T A_k + C_k^T J_{u,k} \quad (3.69)$$

$$J_{s,k}^* = J_{s,k} + A_k^T J_{uu,k} D_k + J_{us,k}^T A_k + D_k^T J_{u,k} \quad (3.70)$$

$$J_{XX,k}^* = J_{XX,k} + B_k^T J_{uu,k} B_k + B_k^T J_{uX,k} + J_{uX,k}^T B_k \quad (3.71)$$

$$J_{\lambda\lambda,k}^* = J_{\lambda\lambda,k} + C_k^T J_{uu,k} C_k + C_k^T J_{u\lambda,k} + J_{u\lambda,k}^T C_k \quad (3.72)$$

$$J_{ss,k}^* = J_{ss,k} + D_k^T J_{uu,k} D_k + D_k^T J_{us,k} + J_{us,k}^T D_k \quad (3.73)$$

$$J_{X\lambda,k}^* = J_{X\lambda,k} + B_k^T J_{uu,k} C_k + B_k^T J_{u\lambda,k} + J_{uX,k}^T C_k \quad (3.74)$$

$$J_{Xs,k}^* = J_{Xs,k} + B_k^T J_{uu,k} D_k + B_k^T J_{us,k} + J_{uX,k}^T D_k \quad (3.75)$$

$$J_{\lambda s,k}^* = J_{\lambda s,k} + C_k^T J_{uu,k} D_k + C_k^T J_{us,k} + J_{u\lambda,k}^T D_k \quad (3.76)$$

At the end of the backward sweep, once the control law has been found at every segment of leg j , the controls-free cost-to-go J_0^* is expanded with respect to \mathbf{X}_0 , $\boldsymbol{\lambda}$, and \mathbf{s} :

$$\begin{aligned} \delta J_0^* &= ER_0 + J_{X,0}^{*T} \delta \mathbf{X}_0 + J_{\lambda,0}^{*T} \delta \boldsymbol{\lambda} + J_{s,0}^{*T} \delta \mathbf{s} \\ &\quad + \frac{1}{2} \delta \mathbf{X}_0^T J_{XX,0}^* \delta \mathbf{X}_0 + \frac{1}{2} \delta \boldsymbol{\lambda}^T J_{\lambda\lambda,0}^* \delta \boldsymbol{\lambda} + \frac{1}{2} \delta \mathbf{s}^T J_{ss,0}^* \delta \mathbf{s} \\ &\quad + \delta \mathbf{X}_0^T J_{X\lambda,0}^* \delta \boldsymbol{\lambda} + \delta \mathbf{X}_0^T J_{Xs,0}^* \delta \mathbf{s} \\ &\quad + \delta \boldsymbol{\lambda}^T J_{\lambda s,0}^* \delta \mathbf{s} \end{aligned} \quad (3.77)$$

Then, once the control law and the expected reduction have been obtained for every leg, the controls-free partials are used in the NLP solver in order to minimize the total cost with respect to the \mathbf{s} vector, and form the necessary sensitivities for the Lagrange multipliers update.

3.3.2 Multi-leg Quadratic Expansions

Once a control law for each leg has been obtained using the method described in [Section 3.3.1](#), the total cost can be expanded in terms of $\delta\mathbf{X}_0^{(j)}$, $\delta\lambda^{(j)}$, and $\delta\mathbf{s}^{(j)}$, $j \in [1, \dots, n_l]$. By definition of the \mathbf{s} vector: $\delta\mathbf{X}_0^{(j)} = \delta\mathbf{s}^{(j)}$ which, when plugged in [Eq. \(3.77\)](#), yields, for leg j :

$$\begin{aligned}\delta J_0^{*(j)} &= ER_0^{(j)} + J_{X,0}^{*(j)T} \delta\mathbf{s}^{(j)} + J_{\lambda,0}^{*(j)T} \delta\boldsymbol{\lambda}^{(j)} + J_{s,0}^{*(j)T} \delta\mathbf{s} \\ &\quad + \frac{1}{2} \delta\mathbf{s}^{(j)T} J_{XX,0}^{*(j)} \delta\mathbf{s}^{(j)} + \frac{1}{2} \delta\boldsymbol{\lambda}^{(j)T} J_{\lambda\lambda,0}^{*(j)} \delta\boldsymbol{\lambda}^{(j)} + \frac{1}{2} \delta\mathbf{s}^T J_{ss,0}^{*(j)} \delta\mathbf{s} \\ &\quad + \delta\mathbf{s}^{(j)T} J_{X\lambda,0}^{*(j)} \delta\boldsymbol{\lambda}^{(j)} + \delta\mathbf{s}^{(j)T} J_{Xs,0}^{*(j)} \delta\mathbf{s} \\ &\quad + \delta\mathbf{s}^T J_{s\lambda,0}^{*(j)} \delta\boldsymbol{\lambda}^{(j)}\end{aligned}\tag{3.78}$$

Note the dimensions of the different matrices: $\mathbf{s}^{(j)} \in \mathbb{R}^{n_X}$, $\mathbf{s} \in \mathbb{R}^{n_X \times n_l}$, $\boldsymbol{\lambda}^{(j)} \in \mathbb{R}^{n_\lambda^{(j)}}$, and the gradients and Hessians have corresponding dimensions. The expansion of the total cost is:

$$\delta J = \sum_{j=1}^{n_l} \delta J_0^{*(j)}\tag{3.79}$$

Analogous to the \mathbf{s} vector (see [Eq. \(3.14\)](#)), define the $\boldsymbol{\Lambda}$ vector as:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \boldsymbol{\lambda}^{(1)} \\ \boldsymbol{\lambda}^{(2)} \\ \vdots \\ \boldsymbol{\lambda}^{(n_l)} \end{bmatrix}\tag{3.80}$$

and rewrite [Eq. \(3.79\)](#) as:

$$\begin{aligned}\delta J &= \sum_{j=1}^{n_l} [ER_0^{(j)}] + J_S^T \delta\mathbf{s} + J_\Lambda^T \delta\boldsymbol{\Lambda} \\ &\quad + \frac{1}{2} \delta\mathbf{s}^T J_{SS} \delta\mathbf{s} + \frac{1}{2} \delta\boldsymbol{\Lambda}^T J_{\Lambda\Lambda} \delta\boldsymbol{\Lambda} \\ &\quad + \delta\mathbf{s}^T J_{S\Lambda} \delta\boldsymbol{\Lambda}\end{aligned}\tag{3.81}$$

where: J_S and J_{SS} are the Jacobian and Hessian of the total cost with respect to the \mathbf{s} vector, and are formed as such:

$$J_S = \begin{bmatrix} J_{X,0}^{*(1)} \\ J_{X,0}^{*(2)} \\ \vdots \\ J_{X,0}^{*(n_l)} \end{bmatrix} + \sum_{j=1}^{n_l} J_{s,0}^{*(j)} \quad (3.82)$$

$$J_{SS} = \text{diag}(J_{XX,0}^{*(1)}, J_{XX,0}^{*(2)}, \dots, J_{XX,0}^{*(n_l)}) + \begin{bmatrix} J_{Xs,0}^{*(1)} \\ J_{Xs,0}^{*(2)} \\ \vdots \\ J_{Xs,0}^{*(n_l)} \end{bmatrix} + \begin{bmatrix} J_{sX,0}^{*(1)} & J_{sX,0}^{*(2)} & \dots & J_{sX,0}^{*(n_l)} \end{bmatrix} + \sum_{j=1}^{n_l} J_{ss}^{*(j)} \quad (3.83)$$

where, if M_1, \dots, M_n are $n \cdot m \times p$ matrices, $\text{diag}(M_1, \dots, M_n)$ is the $n \cdot m \times n \cdot p$ non-square block diagonal matrix defined as:

$$\text{diag}(M_1, \dots, M_n) = \begin{bmatrix} M_1 & \mathbf{0}_{m \times p} & \dots & \mathbf{0}_{m \times p} \\ \mathbf{0}_{m \times p} & M_2 & \dots & \mathbf{0}_{m \times p} \\ \vdots & \dots & \dots & \dots \\ \mathbf{0}_{m \times p} & \mathbf{0}_{m \times p} & \dots & M_n \end{bmatrix} \quad (3.84)$$

J_Λ and $J_{\Lambda\Lambda}$ are the Jacobian and Hessian of the total cost with respect to the vector Λ formed of all the Lagrange multipliers. They can be expressed as:

$$J_\Lambda = \begin{bmatrix} J_{\lambda,0}^{*(1)} \\ J_{\lambda,0}^{*(2)} \\ \vdots \\ J_{\lambda,0}^{*(n_l)} \end{bmatrix} \quad (3.85)$$

$$J_{\Lambda\Lambda} = \text{diag}(J_{\lambda\lambda,0}^{*(1)}, J_{\lambda\lambda,0}^{*(2)}, \dots, J_{\lambda\lambda,0}^{*(n_l)}) \quad (3.86)$$

Finally, the cross derivatives matrix $J_{S\Lambda}$ is formed as such:

$$J_{S\Lambda} = \text{diag}(J_{X\lambda,0}^{*(1)}, J_{X\lambda,0}^{*(2)}, \dots, J_{X\lambda,0}^{*(n_l)}) + \begin{bmatrix} J_{s\lambda,0}^{*(1)} & J_{s\lambda,0}^{*(2)} & \dots & J_{s\lambda,0}^{*(n_l)} \end{bmatrix} \quad (3.87)$$

In Eqs. (3.82)-(3.87), the subscripts X , s , and λ indicate a partial derivatives with respect to the initial state of the leg (dimension n_X), the initial conditions of all of the legs (dimension $n_x \times n_l$), and the Lagrange multipliers of the leg (dimension $n_{\lambda(j)}$) respectively.

As in classical augmented Lagrangian algorithms [22, 75], the inner loop must be solved (or approximately solved) using an unconstrained optimizer, before updating the Lagrange multipliers. Therefore, the multiple-shooting variable \mathbf{s} has to be updated first, then a control law with feedback in $\boldsymbol{\lambda}$ can be obtained. In the current MDDP formulation, the trust-region algorithm is used to minimize Eq. (3.81) with respect to $\delta\mathbf{s}$ (see Section 3.4.1), and returns a shifted Hessian matrix \tilde{J}_{SS} , which is used in the new control law:

$$\delta\mathbf{s} = -\tilde{J}_{SS}^{-1}(J_S + J_{S\Lambda}\delta\boldsymbol{\Lambda}) \quad (3.88)$$

$$\delta\mathbf{s} = A_S + C_S\delta\boldsymbol{\Lambda} \quad (3.89)$$

with

$$A_S = -\tilde{J}_{SS}^{-1}J_S \quad (3.90)$$

$$C_S = -\tilde{J}_{SS}^{-1}J_{S\Lambda} \quad (3.91)$$

The developments presented up to Eq. (3.91) define the algorithm's inner loop, which is iterated before the Lagrange multipliers update. Therefore, in most iterations of the inner loop $\delta\boldsymbol{\Lambda} = \mathbf{0}$ and the control law of Eq. (3.89) becomes a simple update. On the first iteration after a Lagrange multipliers update, the feedback C_S is utilized.

3.3.3 Lagrange Multipliers Update

As mentioned in [Section 3.2.1](#), various multipliers updates appear in the literature, and most authors prefer first-order updates, as the computation of higher-order partials of the augmented cost with respect to the multipliers can be computationally expensive. However, because of the STM-based approach of MDDP, the second-order partials of the augmented cost with respect to Λ can be computed without additional integrations, by using the mapping and updating presented in [Section 3.3.1](#) and [Section 3.3.2](#). Therefore, a second-order update is implemented. First, the control law for δs is plugged in the expansion of [Eq. \(3.81\)](#) to obtain the s -free cost-to-go expansion and update equations:

$$\delta J = \hat{E}R + \hat{J}_\Lambda^T \delta \Lambda + \frac{1}{2} \delta \Lambda^T \hat{J}_{\Lambda\Lambda} \delta \Lambda \quad (3.92)$$

with:

$$\hat{E}R = \sum_{j=1}^{n_l} ER_0^{(j)} + ER_s \quad (3.93)$$

$$= \sum_{j=1}^{n_l} ER_0^{(j)} + J_S^T A_S + \frac{1}{2} A_S^T J_{SS} A_S \quad (3.94)$$

$$\hat{J}_\Lambda = J_\Lambda + A_S^T J_{SS} C_S + J_{S\Lambda}^T A_S + C_S^T J_S \quad (3.95)$$

$$\hat{J}_{\Lambda\Lambda} = J_{\Lambda\Lambda} + C_S^T J_{SS} C_S + C_S^T J_{S\Lambda} + J_{S\Lambda}^T C_S \quad (3.96)$$

$$(3.97)$$

Then, the trust-region is used one more time to obtain the update law:

$$\delta \Lambda = \tilde{\hat{J}}_{\Lambda\Lambda}^{-1} \hat{J}_\Lambda \quad (3.98)$$

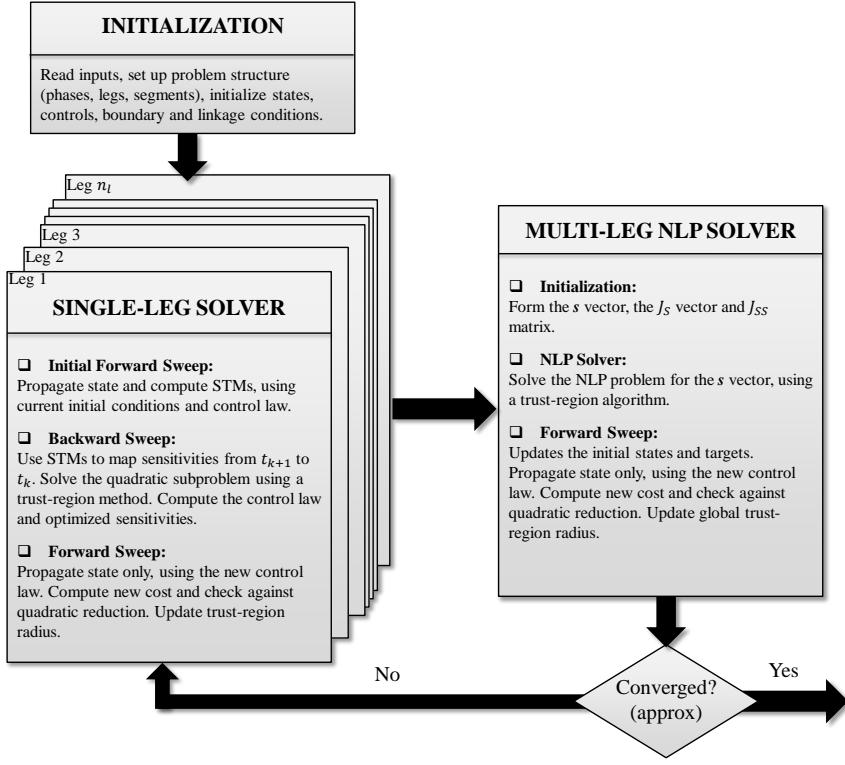


Figure 3.2: Inner loop of the MDDP algorithm

and the final expected reduction:

$$ER_f = \hat{ER} + ER_\lambda \quad (3.99)$$

$$= \hat{ER} - \hat{J}_\Lambda^T \delta \Lambda - \frac{1}{2} \delta \Lambda^T \hat{J}_{\Lambda \Lambda} \delta \Lambda \quad (3.100)$$

The mathematical developments necessary to the solution of both the inner and outer loops have now been presented, and Section 3.4 shows their use in the current implementation of the MDDP algorithm.

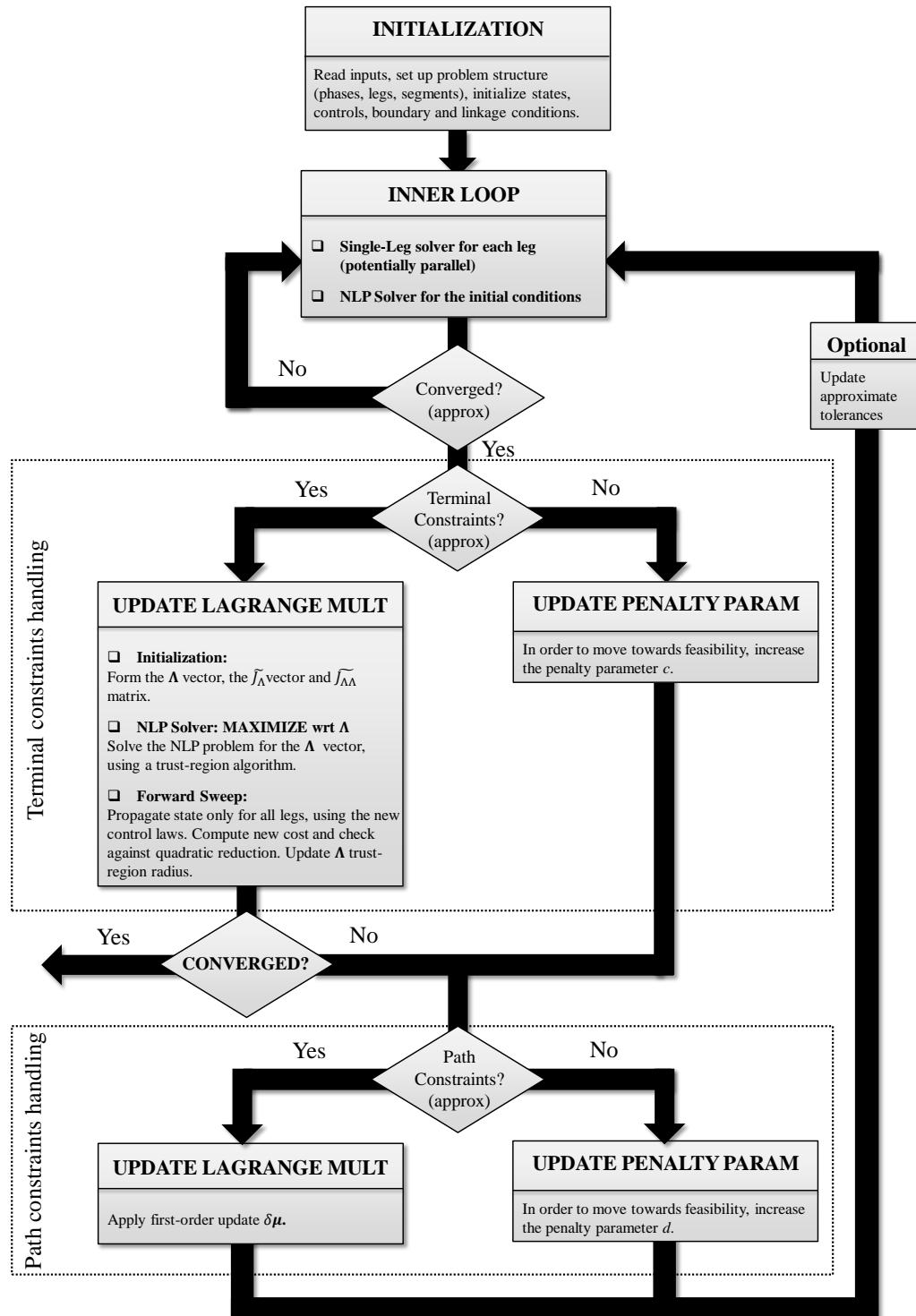


Figure 3.3: Structure of the MDDP algorithm

3.4 Overview of the MDDP algorithm

The MDDP algorithm is structured following the same primal-dual principles as classical augmented Lagrangian methods. An inner loop solves for the optimal controls of each leg independently as well as for the optimal initial conditions of all legs, effectively evaluating the dual functional and its partials for fixed values of the penalty parameter and Lagrange multipliers. An outer loop then maximizes the dual functional with respect to the Lagrange multipliers introduced in [Section 3.2](#). This section presents the main steps taken in solving the inner and outer loop using MDDP. [Figure 3.2](#) and [Fig. 3.3](#) summarize those steps.

The first task is the initialization of the problem structure. As mentioned in [Section 3.1](#), the problem is initially formulated with a multiple-phase structure. It is therefore necessary to create the legs for each phase, to associate them with the correct dynamics, costs and constraints, and to initialize the controls vectors, initial states, and boundary and linkage conditions for each leg.

Then, the inner loop of the algorithm is entered, which solves for the controls of each leg in the problem (using the modified HDDP algorithm) and their initial conditions (using the multi-leg NLP). Because of the multiple-shooting formulation, the legs are independent from each other for the “single-leg solver”, which updates the control laws. The steps marked with an asterisk (*) can be performed in parallel over all the legs, as illustrated in [Fig. 3.2](#). An iteration of the inner loop is organized as follows:

- **Leg initial forward sweep** (*): An initial forward sweep propagates the state and computes the STMs using the current initial state and control law. In the current implementation of MDDP, the STMs can be obtained using one of three techniques, which are explained and compared in [Chapter 5](#): the propagation of the variational equations, finite differences methods, or multi-complex step derivatives [181]. Moreover, it is possible to reduce some of the complexity and computational labor by using Quasi-Newton approximations to the second-order STM, as detailed in [Section 3.5](#).
- **Leg backward sweep** (*): To initialize the backward sweep, the augmented cost function $J^{(j)}$ of [Eq. \(3.43\)](#) is evaluated and its partials are obtained, either analytically when possible, or using finite differences or complex-step derivation. For $k = n^{(j)}$ to 1, the cost-to-go partials are mapped backwards using [Eqs. \(3.51\)-\(3.59\)](#). A trust-region algorithm (see [Section 3.4.1](#)) is used to compute the inverse of the shifted Hessian \tilde{J}_{uu}^{-1} , and the control law is formed using [Eqs. \(3.62\)-\(3.66\)](#). The optimized, controls-free partials and the expected reduction are computed with [Eqs. \(3.67\)-\(3.76\)](#).
- **Leg convergence test** (*): See [Section 3.4.3](#).
- **Leg forward sweep** (*): In order to ensure that the quadratic approximations are valid, the state only is propagated using the new control law (keeping $\mathbf{X}_0^{(j)}$ the same), the leg's cost ratio is computed and used to accept or reject the iteration, and update the trust-region radius $\Delta_u^{(j)}$.

If the iteration is unacceptable, the newly computed control law is dismissed, the trust-region radius is reduced, and a new leg backward sweep starts from the reference trajectories computed during the initial forward sweep, as explained in [Section 3.4.2](#)

- **All legs iteration check:** The total effective reduction is compared to the total expected reduction, and if the iteration is deemed unreliable, the $\Delta_u^{(j)}$ radii of all legs is reduced before restarting all legs backward sweeps. Note that there is no need for a new forward sweep for this step, as the new costs of each leg have been computed during the leg forward sweep.
- **Update of the multiple-shooting variables:** The global sensitivities J_S, J_{SS} are formed using [Eq. \(3.82\)](#) and [Eq. \(3.83\)](#). The trust-region algorithm is used to solve for δs .
- **Inner loop convergence test:** See [Section 3.4.3](#)
- **Initial conditions forward sweep:** A new forward sweep propagates all legs with their new initial conditions and the recently-accepted leg control laws. The cost ratio is computed, the trial iterate is accepted or rejected, the trust-region radius Δ_s is updated (see [Section 3.4.2](#)). If the iteration is unacceptable, the δs is dismissed, and the trust-region is applied again to update the multiple-shooting variables, with a reduced radius.

In the spirit of Augmented Lagrangian algorithms such as that of Ref. [75], this

iteration is repeated until *approximate* convergence is reached. The outer loop can then proceed to update the penalty parameters or Lagrange multipliers. Because the path constraints and terminal constraints are treated differently (see [Section 3.2.3](#)), the outer loop updates them separately, starting with the terminal constraints:

- **Determine violation of the terminal constraints**
- **Update penalty parameter:** When the terminal constraints are not approximately satisfied, the penalty parameter is updated according to [Eq. \(3.47\)](#)
- **Update terminal constraints Lagrange multipliers:** If the constraints are approximately satisfied, [Eqs. \(3.85\)-\(3.87\)](#) are used to form the global sensitivities to the Lagrange multipliers. [Equation \(3.91\)](#) is used to compute the feedback matrix C_S , and [Eqs. \(3.94\)-\(3.96\)](#) are used to obtain the \mathbf{s} -free sensitivities \hat{J}_Λ and $\hat{J}_{\Lambda\Lambda}$. The trust-region algorithm is used to maximize the cost with respect to $\boldsymbol{\lambda}$.
- **Outer loop convergence test:** See [Section 3.4.3](#)
- **Lagrange multipliers forward sweep:** If the Lagrange multipliers have been updated, the new trajectory is propagated using the control laws of [Eq. \(3.89\)](#) and [Eq. \(3.62\)](#), the cost ratio is computed and the trust-region radius Δ_λ updated (see [Section 3.4.2](#)). If the update is unreliable, $\delta\boldsymbol{\Lambda}$ is rejected, and the update computed again with a reduced trust-region radius.

The penalty parameter and Lagrange multipliers associated with the path constraints are updated next:

- **Determine violation of the path constraints**
- **Update penalty parameter:** When the path constraints are not approximately satisfied, the penalty parameter is updated according to [Eq. \(3.48\)](#)
- **Update path constraints Lagrange multipliers:** If the constraints are approximately satisfied, the first-order update of [Eq. \(3.46\)](#) is used. The update uses the latest propagation of the trajectory (usually from the initial conditions or Lagrange multipliers forward sweep) as the reference for \mathbf{X}_k , $k \in [1, \dots, n^{(j)}]$.

Note that there is no convergence test or forward sweep after the path constraints multipliers update. The first-order optimality condition with respect to $\boldsymbol{\mu}$ corresponds to satisfaction of the path constraints. The Hessian of the augmented cost with respect to $\boldsymbol{\mu}$ is not needed, and therefore the second-order optimality condition can not be computed. For the same reason, the expected reduction for a change in $\boldsymbol{\mu}$ is not computed, and therefore no new forward sweep is necessary. The path constraints multipliers update is similar to the penalty parameters update in that its magnitude and effects on the augmented objective function are not controlled.

Finally, the last step in an MDDP major iteration is to update the tolerances associated with approximate convergence, as described in [Section 3.4.3](#).

3.4.1 Trust-Region Algorithm

The control laws of Eq. (3.62), Eq. (3.89), and Eq. (3.98) are computed using the cost-to-go gradients and the shifted Hessians $\tilde{J}_{uu,k}$, \tilde{J}_{SS} and $\tilde{J}_{\Lambda\Lambda}$. Any numerical solver capable of providing a positive-definite shifted Hessian could be used for these steps. The trust-region algorithm has been a popular choice for this application in the past [178, 340], because it guarantees positive-definiteness of the shifted Hessian (and therefore a descent direction), while also limiting the magnitude of the variations, and therefore ensuring that the quadratic expansions are reliable (see Section 3.4.2). The algorithm used in the present implementation is based off of Algorithms 7.3.4 and 7.3.6 of Ref. [78], and is used for the optimization of most decision variables (controls, initial conditions, and terminal constraints Lagrange multipliers). The general quadratic subproblem solved by the trust-region method is:

$$\underset{\delta\sigma}{\text{minimize}} \quad J = J_\sigma \delta\sigma + \frac{1}{2} \delta\sigma^T J_{\sigma\sigma} \delta\sigma \quad \text{subject to} \quad \|D_\sigma \delta\sigma\| \leq \Delta_\sigma \quad (3.101)$$

where $\sigma = \mathbf{u}, \mathbf{s}$ or Λ , and D_σ is a scaling matrix. The trust-region subproblem is solved by ways of shifting the Hessian matrix until: 1) it is positive-definite, and 2) the magnitude of the step is no larger than the trust-region radius. The value λ^* of the shift is the Lagrange multiplier of the trust-region constraint for the problem of Eq. (3.101). Note that the quadratic subproblems can have large variations in dimensions, as n_u is usually small, whereas $n_s = n_l \times n_X$ and $n_\Lambda = \sum_{j=1}^{n_l} n_\lambda^{(j)}$ can be orders of magnitude larger, depending on the number of legs considered, and the number of boundary conditions. Therefore, different implementations of the trust-region algorithm are used: the method

employed to update \mathbf{u} uses full eigendecompositions of the Hessian, whereas the initial conditions and Lagrange multipliers are updated using only Cholesky decompositions. The unconstrained trust-region is in conjunction with a null-space method described in Ref. [179] in order to guarantee adherence of the decision variables to hard bounds. The trust-region is solved a first time using the full $J_{\sigma\sigma}$, and the control variables that lie on or passed their bounds are identified, and assigned a non-feedback update that fixes them on their bound. The corresponding rows and columns of the gradient and Hessian are removed, and a new reduced quadratic subproblem is solved. The process is iterated until the reduced trust-region produces a step that does not violate any of the bounds. The resulting gradient $J_{\sigma,r}$ and Hessian $J_{\sigma\sigma,r}$, reduced to the active decision variables only, are used for convergence assessment. Moreover, the shifted $\tilde{J}_{\sigma\sigma}$ Hessian returned by the null-space method contains only the elements of the reduced shifted Hessian $\tilde{J}_{\sigma\sigma,r}$, effectively canceling the feedback laws for the control variables that violated their bounds. In addition, numerical safeguards are implemented to prevent ill-conditioned Hessians from being detrimental to the solver. When the shifted Hessian is ill-conditioned, the directions corresponding to the smallest eigenvalues are removed, so that $\tilde{J}_{\sigma\sigma}^{-1}$ stays well-conditioned, and does not corrupt the subsequent segments.

Note that because the trust-region method is used for three different updates ($\delta\mathbf{u}$, $\delta\mathbf{s}$, and $\delta\boldsymbol{\Lambda}$), each leg has its own controls radius $\Delta_u^{(j)}$, and both the initial conditions solver and the Lagrange multipliers update have their own radius, Δ_s and Δ_Λ , respectively.

3.4.2 Cost Ratio Computation and Trust-Region Update

Since the MDDP algorithm uses quadratic approximations of the augmented cost function, it is necessary to measure the quality of the Taylor series approximation. This quality control is done through the computation of a cost ratio ρ , which compares the expected quadratic reduction to the effective cost reduction after an update, as shown in Eqs. (3.102)-(3.108). The cost ratio allows to control the size of the trust-region radius, and to reject iterations when the radius is such that the quadratic approximations are unreliable, in a manner similar to Ref. [178]. Each leg has its own cost ratio, and therefore its own radius. The cost ratio could be used to determine the sensitivity and nonlinearity of a specific leg, since it represents the quality of the quadratic approximations. A strategy could be devised to refine the length of the legs depending on this cost ratio, but this adaptive mesh refinement is not explored in the current implementation. The cost ratio is computed after each second-order update, starting with each leg forward sweep:

$$\rho_u^{(j)} = \frac{J^{(j),\text{new}} - J^{(j)}}{ER_0^{(j)}} \quad (3.102)$$

with

$$J^{(j),\text{new}} = J^{(j)}(\mathbf{X}_0^{(j)}, \mathbf{u}_{1,\dots,n^{(j)}}^{\text{new}}, \mathbf{s}, \boldsymbol{\lambda}^{(j)}, \boldsymbol{\mu}_{1,\dots,n^{(j)}}^{(j)} c, d) \quad (3.103)$$

and

$$J^{(j)} = J^{(j)}(\mathbf{X}_0, \mathbf{u}_{1,\dots,n^{(j)}}, \mathbf{s}, \boldsymbol{\lambda}^{(j)}, \boldsymbol{\mu}_{1,\dots,n^{(j)}}^{(j)} c, d) \quad (3.104)$$

As an extra safeguard against unreliable quadratic approximations, a cost ratio for the global problem is also computed after the controls for all legs have been

updated:

$$\rho_u = \frac{\sum_{j=1}^{n_l} J^{(j),\text{new}} - \sum_{j=1}^{n_l} J^{(j)}}{\sum_{j=1}^{n_l} ER_0^{(j)}} \quad (3.105)$$

where $J^{(j),\text{new}}$ and $J^{(j)}$ are defined as in Eq. (3.103) and Eq. (3.104), respectively. This cost ratio is only used to accept or reject an iteration, and does not impact the legs' radii if the iteration is acceptable. It can also be computed without a new forward sweep, since the costs $J^{(j),\text{new}}$ have been computed during the legs' individual forward sweeps. A new cost ratio is computed after the initial conditions update:

$$\rho_s = \frac{\sum_{j=1}^{n_l} J^{(j),\text{new}} - \sum_{j=1}^{n_l} J^{(j)}}{\hat{ER}} \quad (3.106)$$

where

$$J^{(j),\text{new}} = J^{(j)}(\mathbf{X}_0^{(j),\text{new}}, \mathbf{u}_{1,\dots,n^{(j)}}^{\text{new}}, \mathbf{s}^{\text{new}}, \boldsymbol{\lambda}^{(j)}, \boldsymbol{\mu}_{1,\dots,n^{(j)}}^{(j)} c, d) \quad (3.107)$$

and $J^{(j)}$ is defined in Eq. (3.104). Finally, the cost ratio for the terminal constraints multipliers update is:

$$\rho_\lambda = \frac{\sum_{j=1}^{n_l} J^{(j),\text{new}} - \sum_{j=1}^{n_l} J^{(j)}}{ER_f} \quad (3.108)$$

where

$$J^{(j),\text{new}} = J^{(j)}(\mathbf{X}_0^{(j),\text{new}}, \mathbf{u}_{1,\dots,n^{(j)}}^{\text{new}}, \mathbf{s}^{\text{new}}, \boldsymbol{\lambda}^{(j),\text{new}}, \boldsymbol{\mu}_{1,\dots,n^{(j)}}^{(j)} c, d) \quad (3.109)$$

and $J^{(j)}$ is defined in Eq. (3.104). Note that in Eq. (3.103), Eq. (3.107), and Eq. (3.109), the “new” superscript does not necessarily denote the same quantity: for instance, \mathbf{u}^{new} is different after each update, since the feedback terms modify the control law.

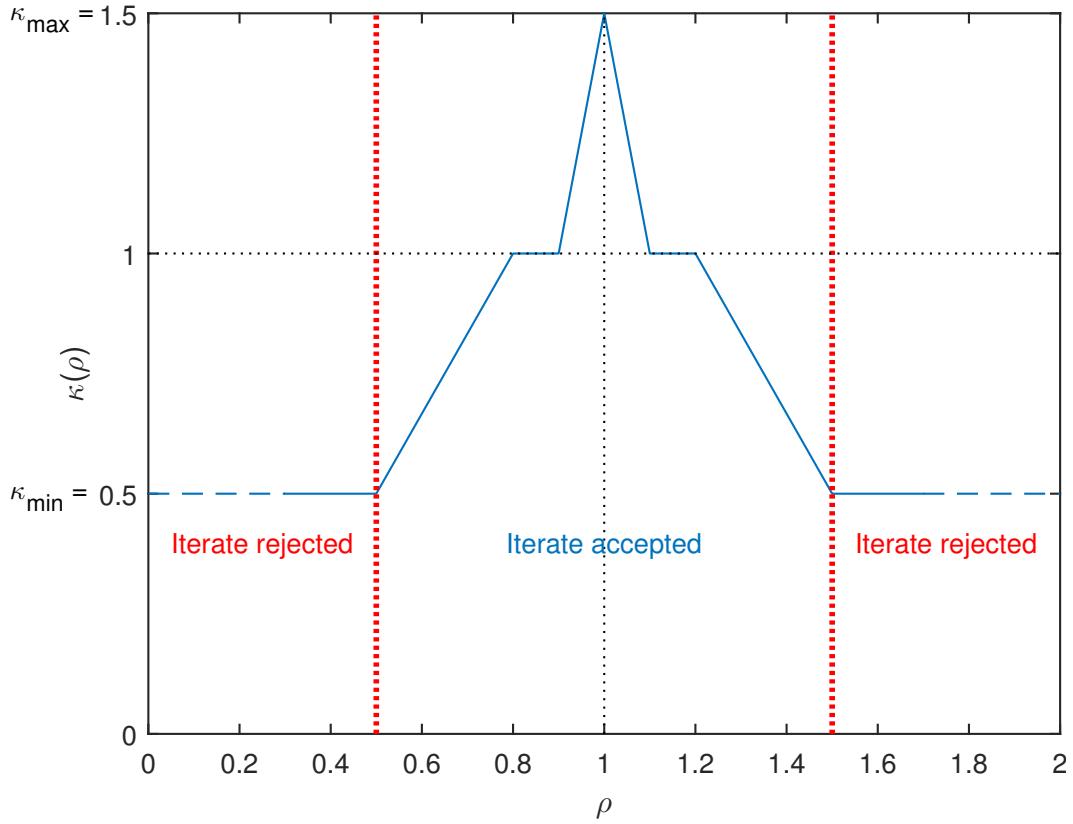


Figure 3.4: Heuristic multiplier $\kappa(\rho)$ for trust-region radius update

Once a cost ratio is computed, it is used for the acceptance of trial iterates: a cost ratio close to 1 indicates that the effective reduction is close to the reduction predicted by the second-order Taylor series approximations. The trial iterate can be accepted, and the algorithm moves on to the next step. When the cost ratio is away from 1 by more than a user defined critical value ϵ_3 , the quadratic approximations did not capture the full change in the objective function, and are deemed unreliable. The trial iterate is therefore rejected, the current parameters are reverted to their state before the most recent quadratic update, and the corresponding trust-region radius is reduced

(see [Fig. 3.4](#)). For the initial conditions or Lagrange multipliers updates, the trust-region algorithm is applied again immediately with the reduced radius. For the controls updates, the full leg backward sweep is started again, using the last initial forward sweep as a reference trajectory. Therefore, the expensive STMs do not have to be computed again. Once a new update is computed, it is submitted to the same cost ratio test.

The cost ratio is also used to heuristically control the trust-region radii for accepted iterations, since it is a scalar metric of the quality of the Taylor series expansions. After every iteration, the trust-region radii is either increased to allow for a larger step to be taken at the next iteration, or decreased to improve the reliability of the quadratic expansions:

$$\Delta_{\sigma,\text{new}} = \kappa(\rho_\sigma) \Delta_\sigma \quad (3.110)$$

where $\sigma = u$, s , or Λ . The heuristic formula for κ in MDDP is based on three critical values ϵ_1 through ϵ_3 , and is described in [Fig. 3.4](#). Note that the values used in [Fig. 3.4](#) ($\epsilon_1 = 0.1$, $\epsilon_2 = 0.2$, $\epsilon_3 = 0.5$) are picked for readability, and are different from the critical values used in practice (see [Section 4.1.2](#)).

3.4.3 Convergence Tests

The final convergence of the algorithm is checked using first- and second-order necessary conditions for optimality. Approximate convergence of the inner loop is to first-order only, and does not include the constraints criteria. Final convergence of leg (j) $\forall j \in [1, \dots, n_l]$ is determined with the following conditions:

1. The expected change in the objective function is small: $ER_0^{(j)} < \epsilon_{\text{conv}}$
2. For every segment, the gradient of the reduced subproblem is sufficiently small (first-order convergence): $\forall k \in [1, \dots, n^{(j)}], \|J_{u,k,r}^{(j)}\| < \epsilon$
3. For every segment, the Hessian of the reduced subproblem is positive-definite (second-order convergence): $J_{uu,k,r}^{(j)} \succ 0, \forall k \in [1, \dots, n^{(j)}]$
4. All path and terminal constraints are satisfied within tolerance: $\mathbf{g}_k^{(j)} \leq \zeta \forall k \in [1, \dots, n^{(j)}]$, and $\psi^{(j)}(\mathbf{X}_{n^{(j)}}^{(j)}, \mathbf{s}) \leq \boldsymbol{\eta}$

The convergence of the inner loop are determined by:

1. Convergence of all the legs following the conditions above
2. The expected change in the objective function is small: $ER_s < \epsilon_{\text{conv}}$
3. The gradient of the reduced subproblem is sufficiently small (first-order convergence): $\|J_{S,r}\| < \epsilon,$
4. The Hessian of the reduced subproblem is positive-definite (second-order convergence): $J_{SS,r} \succ 0$

Convergence of the complete algorithm is subject to:

1. Convergence of the inner loop (all conditions above)
2. The expected change in the objective function is small: $ER_\lambda < \epsilon_{\text{conv}}$
3. The gradient with respect to the Lagrange multipliers is sufficiently small (first-order convergence): $\|J_\Lambda\| < \epsilon$

4. The Hessian with respect to the Lagrange multipliers is negative definite:

$$J_{\Lambda\Lambda} \prec 0$$

Approximate Convergence The approximate convergence of the inner loop is defined as:

1. For all legs, for all segments, the gradient of the reduced subproblem is sufficiently small (first-order convergence): $J_{u,k,r}^{(j)} < \epsilon_{app}, \forall j \in [1, \dots, n_l], \forall k \in [1, \dots, n^{(j)}]$,
2. The gradient of the reduced subproblem is sufficiently small (first-order convergence): $J_{S,r} < \epsilon_{app}$

The approximate satisfaction of the constraints is subject to:

1. For all segments, path constraints are satisfied within approximate tolerance: $\mathbf{g}_k^{(j)} \leq \zeta_{app}^{(j)} \forall k \in [1, \dots, n^{(j)}]$
2. For all legs, terminal constraints are satisfied within approximate tolerance: $\psi^{(j)}(\mathbf{X}_{n^{(j)}}^{(j)}, \mathbf{s}) \leq \eta_{app} \forall j \in [1, \dots, n_l]$

The approximate tolerances are updated at every iteration of the outer loop, according to a scheme adapted from Refs. [75, 230], and described in [Algorithm 1](#) for the end of iteration p .

Algorithm 1 Update of the approximate convergence tolerances

```
1: if  $\psi^{(j)}(\mathbf{X}_{n^{(j)}}^{(j)}, \mathbf{s}) \leq \eta_{\text{app},p} \forall j \in [0, \dots, n_l]$  then
2:    $\epsilon_{\text{app},p+1} \leftarrow \max(\epsilon, (\frac{1}{2c})^\alpha \epsilon_{\text{app},p})$ 
3:    $\eta_{\text{app},p+1} \leftarrow \max(\eta, (\frac{1}{2c})^\beta \eta_{\text{app},p})$ 
4: else
5:    $\epsilon_{\text{app},p+1} \leftarrow \max(\epsilon, (\frac{1}{2c})^\alpha \epsilon_{\text{app},0})$ 
6:    $\eta_{\text{app},p+1} \leftarrow \max(\eta, (\frac{1}{2c})^\beta \eta_{\text{app},0})$ 
7: end if
8: for  $j \leftarrow 1, \dots, n_l$  do
9:   if  $\mathbf{g}_k^{(j)} \leq \zeta_{\text{app},p}^{(j)} \forall k \in [1, \dots, n^{(j)}]$  then
10:     $\zeta_{\text{app},p+1}^{(j)} \leftarrow \max(\zeta^{(j)}, (\frac{1}{2d^{(j)}})^\beta \zeta_{\text{app},p}^{(j)})$ 
11:   else
12:     $\zeta_{\text{app},p+1}^{(j)} \leftarrow \max(\zeta^{(j)}, (\frac{1}{2d^{(j)}})^\beta \zeta_{\text{app},0})$ 
13:   end if
14: end for
```

3.5 Estimating the Second-Order State Transition Matrices

The evaluation of second-order partial derivatives is often the most computationally intense part of an optimization algorithm, as mentioned in [Section 1.2.1.3](#) and extensively described in [Chapter 5](#). [Section 4.3.2](#) show that, for the MDDP algorithm, the second-order STMs computation represents up to 95% of the run time. In order to reduce the execution time of the algorithm, the well-known quasi-Newton Hessian approximations methods are incorporated into MDDP.

3.5.1 Quasi-Newton Methods

Quasi-Newton algorithms are method designed to approximate the second-order order partial derivatives of a nonlinear program. The objective of a typical quasi-Newton algorithm is to find the local minimum of a function $J(\mathbf{X})$. At iteration p , the estimate B_p of the Hessian H of J with respect to \mathbf{X} is updated as such:

$$\mathbf{X}_{p+1} = \mathbf{X}_p + \Delta\mathbf{X}_p \quad (3.111)$$

$$\mathbf{y}_p = J_X(\mathbf{X}_{p+1}) - J_X(\mathbf{X}_p) \quad (3.112)$$

$$B_{p+1} = B_p + U_p \quad (3.113)$$

where \mathbf{X}_p is the value of the unknown vector \mathbf{X} at iteration p , $J_X(\mathbf{X}_p)$ is the gradient of J with respect to \mathbf{X} at that same iteration, and U_p is the matrix update, usually a rank-one or rank-two matrix, depending only on \mathbf{X}_p , $\Delta\mathbf{X}_p$, and \mathbf{y}_p . Quasi-Newton updates are derived from the quadratic expansion of J_X :

$$\mathbf{y}_p = J_X(\mathbf{X}_{p+1}) - J_X(\mathbf{X}_p) = H\Delta\mathbf{X}_p + \mathcal{O}(\Delta X_p^2) \quad (3.114)$$

Therefore, the update is chosen so that it correctly relates \mathbf{y}_p and $\Delta\mathbf{X}_p$:

$$B_{p+1}\Delta\mathbf{X}_p = \mathbf{y}_p \quad (3.115)$$

[Equation \(3.115\)](#) is called the quasi-Newton condition or secant condition for the update. It is unfortunately impossible to extract the full second-order information from the secant equation. Quasi-Newton methods use a symmetric update in order to preserve symmetry of the estimate of the Hessian. Most

updates are rank-one or rank-two matrices [109], $U_p = a\mathbf{r}\mathbf{r}^T$ or $U_p = a\mathbf{r}\mathbf{r}^T + b\mathbf{q}\mathbf{q}^T$, respectively. The quasi-Newton condition yields:

$$(B_p + U_p)\Delta\mathbf{X}_p = \mathbf{y}_p \quad (3.116)$$

$$U_p\Delta\mathbf{X}_p = \mathbf{y}_p - B_p\Delta\mathbf{X}_p \quad (3.117)$$

In the case of a rank-one update:

$$a\mathbf{r}^T\Delta\mathbf{X}_p = \mathbf{y}_p - B_p\Delta\mathbf{X}_p \quad (3.118)$$

Since $a\mathbf{r}^T\Delta\mathbf{X}_p$ is a scalar, \mathbf{r} has to be proportional to $\mathbf{y}_p - B_p\Delta\mathbf{X}_p$. The simplest choice for \mathbf{r} is $\mathbf{r} = \mathbf{y}_p - B_p\Delta\mathbf{X}_p$, resulting in the normalisation condition: $a\mathbf{r}^T\Delta\mathbf{X}_p = 1$, thus defining a . Finally, the rank-one formula is given by:

$$B_{p+1} = B_p + \frac{(\mathbf{y}_p - B_p\Delta\mathbf{X}_p)(\mathbf{y}_p - B_p\Delta\mathbf{X}_p)^T}{(\mathbf{y}_p - B_p\Delta\mathbf{X}_p)^T\Delta\mathbf{X}_p} \quad (3.119)$$

In traditional nonlinear programming, the updated estimate of the Hessian is used in conjunction with the gradient to determine the search direction. It is noted that initial efforts in this study were focused on using QN methods to estimate the Hessian of the performance index with respect to the control variables, as in typical applications. Indeed, Sen and Yakowitz [302] and Rakshit and Sen [271] claim having successfully integrated QN techniques in DDP algorithms. However, in the current study, when QN updates were applied to the performance index directly, the algorithm generally did not perform well. In fact, it can be argued that using QN approximations in this manner is not a well suited strategy for DDP algorithms. As previously discussed (see Section 3.3.1), $J_{uu,k}^{(j)}$ is the Hessian of the augmented performance index with

respect to the controls on the k^{th} segment of leg (j). This Hessian is used in concert with the gradient $J_{u,k}^{(j)}$ to compute a control law for the k^{th} segment on the next iteration. In Refs. [302], [271] and the initial efforts of this study, the QN approximations are utilized to estimate $J_{uu,k}^{(j)}$, based on the current $J_{uu,k}^{(j)}$ estimate and the current and previous iteration u_k and $J_{u,k}^{(j)}$. However, in DDP methods, the gradient and Hessian are computed in a backward sweep and are explicit functions of downstream control laws. They are therefore only accurate for future trajectories that exactly obey their associated downstream control laws. In the DDP architecture, these downstream control laws change each iteration, and therefore one can not expect the QN condition to hold ($J_{uu,k}^{(j)} \Delta u_k = \Delta J_{u,k}^{(j)}$) even when exact second order partials are computed. In essence, unlike classic parameter optimization, in the DDP application, the performance index changes with each iteration and is dependent on downstream control laws that can not be predicted by past iterations. Accordingly, the authors find the application of QN methods in this context tenuous at best. Moreover, using a quasi-Newton approximation directly on the objective function would necessitate to take all Lagrange multipliers into account when computing the second-order estimate, increasing the dimensions of the update.

Alternatively, the QN approximation is well suited for the second-order STM approximations because the defining state dynamics are invariant from iteration to iteration, analogous to the definition of a performance index in parameter optimization, and do not depend on the Lagrange multipliers. Furthermore, MDDP already decouples the computation of the STMs from the main optimization algorithm; therefore, the QN application can be imple-

mented with little to no modification to the existing MDDP algorithm. Thus, in this dissertation, the quasi-Newton update is applied to estimating the second-order STM. This results in a subtle but important modification of the typical algorithm: $\Delta \mathbf{X}_{p+1}$ is not computed using a line search and the descent direction $B_p^{-1} \nabla J(\mathbf{X}_p)$, as it would for a typical quasi-Newton algorithm minimizing a performance index J . Instead, the backward sweep imposes a control update. Then, the forward propagation using the new control law returns a value for $\Delta \mathbf{X}_{p+1}$ which is used in the next iteration of the quasi-Newton estimation. The quasi-Newton algorithm thus has no control over the magnitude of $\Delta \mathbf{X}_p$, which may create observability problems when $\Delta \mathbf{X}_p$ is small. Another important distinction is the frequency and number of required updates. In a classic optimization problem there is typically one function (the performance index) that requires Hessian updates. In the current MDDP application where the second-order STMs are approximated, there are n_x states (and therefore n_x Hessian matrices to estimate), and $n_{\text{seg}} = \sum_{i=0}^m n^{[i]}$ total segments. Each iteration of the single-leg solver requires Hessian updates to $(n_x \times n_{\text{seg}})$ separate functions.

3.5.2 Choosing an Adapted Quasi-Newton Update: the Symmetric Rank 1 Update

Many quasi-Newton updates are documented in the literature, as introduced in [Section 1.2.1.3](#). Among these, the most commonly utilized methods include the Davidon-Fletcher-Powell formula (DFP), the Broyden-Fletcher-Goldfarb-Shanno update (BFGS) or the modified or “damped” BFGS update

from Powell [109]. These updates are well suited for use in generic parameter optimization, partly because they constrain the estimate of the Hessian to be positive-definite, thus ensuring that a descent direction is selected. Furthermore, it is also common to estimate the inverse of the Hessian in order to avoid solving a system of linear equations.

However, in this work, the update is applied to the second-order trajectory STMs and not to the Hessian of the objective function. There is therefore no need to maintain positive-definiteness. In fact, the main criteria for choosing an update is the accuracy of the approximation compared to the true Hessian, making any method preserving positive-definiteness overly constrained. Of course, symmetry is still a desired property for the update, as any Hessian will be symmetric. According to Conn, Gould and Toint [73, 76], the Symmetric Rank 1 (SR1) update generates more accurate Hessian approximations than other updates for nonquadratic functions. Moreover, it has been demonstrated that the estimates generated by the SR1 update do not require an exact line search to be accurate [109]. Not requiring a line search is important for the application of a quasi-Newton update to MDDP since $\Delta \mathbf{X}_p$ is not computed as in traditional methods. The aforementioned reasons make the SR1 update the logical choice for an application to MDDP. The simple update was derived in [Section 3.5.1](#) and is restated here:

$$B_{p+1} = B_p + \frac{\mathbf{r}_p \mathbf{r}_p^T}{\mathbf{r}_p^T \Delta \mathbf{X}_p} \quad (3.120)$$

where $\mathbf{r}_p = \mathbf{y}_p - B_p \Delta \mathbf{X}_p$.

One drawback of the SR1 update is the possibility for the denominator to

vanish, or to be close to vanishing. To prevent numerical divergence in those cases, the following condition has to be respected:

$$\|\mathbf{r}_p^T \Delta \mathbf{X}_p\| \geq \epsilon_{SR1} \|\mathbf{r}_p\| \|\Delta \mathbf{X}_p\| \quad (3.121)$$

where ϵ_{SR1} is a constant taken between 0 and 1. In practice, for the problems considered in this study, ϵ_{SR1} is set to 0.01. If the condition of Eq. (3.121) does not stand, the previous estimate is reused as is: $B_{p+1} = B_p$.

The quasi-Newton update is applied at each iteration of the MDDP algorithm, during the leg forward sweep, when the state and the first-order STM are updated. The augmented state \mathbf{X}_k and the first-order STM Φ_k^1 are computed for each stage k . For iteration p , for the i^{th} element of the state, the update is applied as follows:

$$\mathbf{y}_{k,p,i} = \Phi_{k,p+1}^1(i,:) - \Phi_{k,p}^1(i,:) \quad (3.122)$$

$$\mathbf{r}_{k,p,i} = \mathbf{y}_{k,p,i} - \Phi_{k,p}^2(i,:,:)\Delta \mathbf{X}_{k,p} \quad (3.123)$$

$$\Phi_{k,p+1}^2(i,:,:,:) = \Phi_{k,p}^2(i,:,:,:) + \frac{\mathbf{r}_{k,p,i}\mathbf{r}_{k,p,i}^T}{\mathbf{r}_{k,p,i}^T \Delta \mathbf{X}_{k,p}} \quad (3.124)$$

The quasi-Newton update is applied n_x times for every segment at every iteration. It is not applied on the control vector, since $\dot{u} = 0$, and therefore $\Phi^2(i,:,:,:) = 0_{n \times n}$ when the i^{th} element is part of the control vector. In the following, the MDDP algorithm with quasi-Newton second-order sensitivities is dubbed QMDDP. Chapter 4 contains the application of the MDDP and QMDDP algorithms to various optimal control problems, and an analysis of the performance of the quasi-Newton approximations.

Chapter 4

Applications of the Multiple-Shooting Differential Dynamic Algorithm

The implementation of the MDDP algorithm presented in [Chapter 3](#) is now applied to constrained nonlinear optimal control problems, in order to validate the multiple-shooting quadratic expansions and the Augmented Lagrangian formulation, and to examine the performance of the optimal control solver¹. First, the collection of test cases used in this dissertation is presented. The MDDP algorithm is applied to diverse optimal control applications, from the original Brachistochrone problem to spacecraft trajectory design. The quadratic expansions and update equations of [Section 3.3](#) are validated, and examples of the behavior of the algorithm with equality and inequality path

¹Work from this chapter was presented as:

- **Etienne Pellegrini** and Ryan P. Russell, A Multiple-Shooting Differential Dynamic Programming Algorithm, Paper AAS 17-453, *AAS/AIAA Space Flight Mechanics Meeting*, San Antonio, TX, February 2017.
- **Etienne Pellegrini** and Ryan P. Russell, Quasi-Newton Differential Dynamic Programming for Robust Low-Thrust Optimization, *AIAA/AAS Astrodynamics Specialists Conference*, Minneapolis, MN, August 2012, [doi:10.2514/6.2012-4425](https://doi.org/10.2514/6.2012-4425)

Work from this chapter will be presented as:

- **Etienne Pellegrini** and Ryan P. Russell, Applications of the Multiple-Shooting Differential Dynamic Programming Algorithm with Path and Terminal Constraints, Paper AAS 17-788 *AAS/AIAA Astrodynamics Specialists Conference*, Stevenson, WA, August 2017.

and terminal constraints are given. The performance of the multiple-shooting formulation is evaluated on several trajectory optimization problems, and the use of quasi-Newton approximations for the second-order STM is examined and demonstrated to reduce the computation time for problems with a moderate or high complexity.

4.1 MDDP Algorithm Testing Framework

This section introduces the application cases used for the analysis of the MDDP algorithm, as well as a number of parameters and settings given for reproducibility purposes.

4.1.1 A Collection of Optimal Control Problems

4.1.1.1 Quadratic-Linear Problem

A simple problem with quadratic objective function and linear constraints is used to validate the quadratic expansions and update equations.

The problem is taken from Ref. [179], and is described by Eqs. (4.1)-(4.4).

$$\underset{\mathbf{u}}{\text{minimize}} \quad J_1 = l_N \tag{4.1}$$

$$\text{subject to} \quad \begin{bmatrix} \mathbf{r}_{k+1} \\ \mathbf{v}_{k+1} \\ l_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_k + \mathbf{v}_k \\ \mathbf{v}_k + \mathbf{u}_k \\ l_k + \|\mathbf{u}_k\| \end{bmatrix} \tag{4.2}$$

$$\text{and} \quad \mathbf{r}_N = [1, -1, 0]^T \tag{4.3}$$

$$\text{with I.C.} \quad \{ \mathbf{r}_0 = [1, 1, 1]^T, \mathbf{v}_0 = [1, 1, 1]^T \tag{4.4}$$

where $\mathbf{r}_k \in \mathbb{R}^3$ (LU) is the position vector at segment k , $\mathbf{v}_k \in \mathbb{R}^3$ (LU/TU) is the velocity vector at the same instant, $\mathbf{u}_k \in \mathbb{R}^3$ (LU/TU²) is the control

vector and is akin to an acceleration. The impulse is equivalent to an Euler integration of the equations of motion with $\Delta t = 1$ TU.

4.1.1.2 Brachistochrone problem

The Brachistochrone problem is the original optimal control problem, published by Bernouilli in 1696. The problem statement is:

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

The Brachistochrone is a minimum-time optimal control problem that can be described by Eqs. (4.5)-(4.8):

$$\underset{u}{\text{minimize}} \quad J_1 = t_f \quad (4.5)$$

$$\text{subject to} \quad \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} v \sin(u) \\ v \cos(u) \\ g \cos(u) \end{bmatrix} \quad (4.6)$$

$$\text{and} \quad x(t_f) = x_b, \quad y(t_f) = y_b \quad (4.7)$$

$$\text{with I.C.} \quad \begin{cases} t_0 = 0, & x(t_0) = x_a \\ y(t_0) = y_a, & v(t_0) = 0 \end{cases} \quad (4.8)$$

where (x, y) (LU) are the coordinates of the point, v (LU/TU) is its velocity, and u (rad) is the angle between the direction of movement and the vertical.

4.1.1.3 Van Der Pol Oscillator

The Van Der Pol oscillator is a classical nonlinear second-order system often used for the validation and analysis of optimal control algorithms [42],

205, 81]. The problem is described by Eqs. (4.9)-(4.12).

$$\underset{u}{\text{minimize}} \quad J_1 = x_3(t_f) \quad \text{or} \quad J_2 = t_f \quad (4.9)$$

$$\text{subject to} \quad \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} (1 - x_2^2)x_1 - x_2 + u \\ x_1 \\ \frac{1}{2}(x_1^2 + x_2^2 + u^2) \end{bmatrix} \quad (4.10)$$

$$\text{and} \quad \begin{cases} x_2(t_f) - x_1(t_f) + d_f = 0, & u_l \leq u \leq u_u \\ x_1 \geq x_{1,l}, & x_2 \geq x_{2,l} \end{cases} \quad (4.11)$$

$$\text{with I.C.} \quad \begin{cases} t_0 = 0, & t_f = 6 \\ \mathbf{x}(t_0) = [1, 1, 0] \end{cases} \quad (4.12)$$

4.1.1.4 One-Dimensional Landing Problem

The one-dimensional landing problem of Ref. 175 is considered. The thrust appears linearly in the Hamiltonian, resulting in a bang-bang optimal control.

$$\underset{u}{\text{minimize}} \quad J_1 = -m(t_f) \quad (4.13)$$

$$\text{subject to} \quad \begin{bmatrix} \dot{x} \\ \dot{v} \\ \dot{m} \end{bmatrix} = \begin{bmatrix} v \\ -g + \frac{u}{m} \\ \frac{-u}{g_0 I_{SP}} \end{bmatrix} \quad (4.14)$$

$$\text{and} \quad \begin{cases} x(t_f) = 0, & v(t_f) = 0 \\ u_l \leq u \leq u_u \end{cases} \quad (4.15)$$

$$\text{with I.C.} \quad \begin{cases} t_0 = 0, & x(t_0) = 1 \\ v(t_0) = -0.783, & m(t_0) = 1 \\ g_0 I_{SP} = 2.349, & g = 1 \end{cases} \quad (4.16)$$

where x (LU) is the height of the spacecraft measured from the ground, v (LU/TU) is its velocity, and m (MU) its mass. The control vector is u (MU LU/TU²), which is the vehicle thrust. The final time t_f (TU) is left free. Note that the cost J_1 is the opposite of the final mass, since the goal is to maximize the final mass, and MDDP is written to minimize an objective function.

4.1.1.5 Two-Dimensional Orbit Transfer

A planar orbit transfer problem similar to that from Bryson and Ho [56, pp. 66-69] is considered. In order to allow for coast arcs, the problem is slightly modified to aim for a fixed radius while minimizing the consumption of fuel or the quadratic integral of the thrust, rather than searching for the largest radius possible in a fixed final time.

$$\underset{T,\theta}{\text{minimize}} \quad J_1 = -x_5(t_f) \quad \text{or} \quad J_2 = x_6(t_f) \quad (4.17)$$

subject to

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_3 \\ \frac{x_4}{x_1} \\ \frac{x_4^2}{x_1} - \frac{\mu}{x_1^2} + \frac{T}{x_5} \sin \theta \\ -\frac{x_3 x_4}{x_1} + \frac{T}{x_5} \cos \theta \\ -T/(g_0 I_{SP}) \\ \frac{1}{2} T^2 \end{bmatrix} \quad (4.18)$$

and

$$\begin{cases} u_l \leq u \leq u_u, & x_1(t_f) = r_f \\ x_3(t_f) = 0, & x_4(t_f) = \sqrt{\frac{\mu}{r_f}} \end{cases} \quad (4.19)$$

with I.C. $\{ t_0 = 0, \mathbf{x}(t_0) = [1, 0, 0, 1, 1, 0] \}$ (4.20)

where x_1 (LU) is the radial distance, x_2 (rad) the polar angle, x_3 (LU/TU) and x_4 (rad/TU) their derivatives, x_5 (MU) the mass, x_6 (LU^2/TU^3) is the integral of the square of the thrust, T (LU/TU²) the thrust magnitude, and θ (rad) the angle of the thrust with respect to the tangential direction. The gravitational parameter is $\mu = 1$ (LU³/TU²), and $g_0 I_{SP} = 5$ (LU/TU). Note that the J_2 objective function is a quadratic function of the thrust, yielding a smoother solution than the typical minimum-fuel problem optimizing J_1 , which results in bang-bang controls.

4.1.1.6 Three-Dimensional Two-Body Problem

A Cartesian representation of the three-dimensional two-body problem with piecewise constant parameterization of the controls is used to model low-thrust spacecraft trajectories. The problem is described by Eqs. (4.21)-(4.24):

$$\underset{\mathbf{T}}{\text{minimize}} \quad J_1 = -m(t_f) \quad \text{or} \quad J_2 = x_8(t_f) \quad (4.21)$$

subject to
$$\begin{bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \\ \dot{m} \\ \dot{x}_8 \end{bmatrix} = \begin{bmatrix} -\frac{\mu}{r^3} \mathbf{r} + \frac{\mathbf{T}}{m} + \mathbf{a}_{\text{pert}} \\ -\frac{g_0 I_{SP}}{T^2} \\ \frac{T}{2} \end{bmatrix} \quad (4.22)$$

and
$$\begin{cases} u_l \leq T = \|\mathbf{T}\| \leq u_u, & \mathbf{u}_l \leq \mathbf{T} \leq \mathbf{u}_u \\ r(t_f) = \|\mathbf{r}(t_f)\| \geq r_f, & \mathbf{r}(t_f)^T \mathbf{v}(t_f) = 0 \end{cases} \quad (4.23)$$

with I.C. $\{ t_0 = 0, \mathbf{x}(t_0) = [1, 0, 0, 0, 0, 1, 0, 1, 0] \} \quad (4.24)$

where $\mathbf{r} \in \mathbb{R}^3$ (LU) is the Cartesian position of the spacecraft, $\mathbf{v} \in \mathbb{R}^3$ (LU/TU) its velocity, m (MU) is the mass, $\mathbf{T} \in \mathbb{R}^3$ (LU/TU²) is the thrust, and $\mathbf{a}_{\text{pert}} \in \mathbb{R}^3$ (LU/TU²) is a perturbing acceleration, which can be used to include the dominant spherical harmonics term J_2 for instance. As for the two-dimensional spacecraft problem, the quadratic integral of the thrust x_8 (LU²/TU³) is added to the state vector and results in a smoother optimal control problem when used as cost. It also is a cost of Lagrange converted into the Mayer form, demonstrating the validity of the augmentation of the state vector explained in Section 3.1. Note that the constraints imposed on the controls in Eq. (4.23) are different by nature: the vector constraints $\mathbf{u}_l \leq \mathbf{T} \leq \mathbf{u}_u$ are handled using the null-space trust-region method whereas the norm constraint $u_l \leq T = \|\mathbf{T}\| \leq u_u$ is satisfied using the Augmented Lagrangian approach for path inequality constraints. A number of different initial and

final conditions, as well as different path and terminal constraints are utilized in the present chapter, and will be introduced alongside their solutions in the following sections.

4.1.2 Settings & Parameters

For reproducibility purposes, the main tuning parameters and their values used for the applications in this chapter are listed in [Table 4.1](#). It is important to keep in mind that, within reasonable limits, the exact parameters should not have significant influence on the robustness of the convergence and the final solution found by the algorithm. The tuning of these parameters primarily influences the rate of convergence. The values listed below are the default used for all problems, except where noted. However, different tunings have been tested to check robustness of the algorithm with respect to the parameters. Unless noted otherwise, the propagation of the EOMs is accomplished using a fixed-step Runge-Kutta-Fehlberg of order 8 (RKF8). One integration step is taken per segment. The partial derivatives are obtained using multi-complex step differentiation.

All simulations in this study use the Intel Visual Fortran Compiler v.17.0.0.109 on a Windows 7 workstation running on a single core of a quad-core Intel Xeon W3550 CPU with 3.07GHz clock speed, and 24GB of RAM.

4.2 Validation of the MDDP algorithm

The examples presented in this section use the problems of [Section 4.1.1](#) to validate the MDDP algorithm described in [Chapter 3](#). The quadratic ex-

Table 4.1: Tuning parameters and their default values

Parameter	Description	Value
$(\epsilon_1, \epsilon_2, \epsilon_3)$	See Section 3.4.2	$(0.01, 0.1, 0.5)$
$(\kappa_{\min}, \kappa_{\max})$	See Section 3.4.2	$(0.5, 1.5)$
$(sg_B, sg_C, sg_D, sg_{Cs})$	Safeguards on the feedback matrices B_k, C_k, D_k , and C_s , as described in Ref. [179]	10^{10}
$(sg_{\text{controls}}, sg_{\delta s})$	Safeguards on the control update, as described in Ref. [179] , and similar safeguard on δs	10^3
(γ_c, γ_d)	See Eq. (3.47) and Eq. (3.48)	$(1.2, 1.2)$
(α, β)	See Algorithm 1	$(0.2, 0.1)$
ϵ_{conv}	See Section 3.4.3	10^{-10}
$(\epsilon, \epsilon_{\text{app},0})$	See Section 3.4.3	$(10^{-6}, 10^{-2})$
$(\eta, \eta_{\text{app},0})$	See Section 3.4.3	$(10^{-8}, 10^{-2})$
$(\zeta, \zeta_{\text{app},0})$	See Section 3.4.3	$(10^{-8}, 10^{-2})$

pansions and update equations are verified, and the behavior of the algorithm subject to various equality and inequality path and terminal constraints is examined. The single-shooting version of the MDDP algorithm (using a single phase and leg) is employed in this section. The single-shooting version of MDDP includes the single-leg solver as well as the update to the initial conditions of the single leg and Lagrange multipliers. Examples are shown for a varied subset of the problems of [Section 4.1.1](#) in order to demonstrate the capability of the MDDP algorithm to solve general optimal control problems.

4.2.1 Validation of the Quadratic Expansions and Updates for the Single-Leg Solver

The expansions and update equations of [Section 3.3.1](#) reduce a nonlinear optimal control problem to a sequence of quadratic programs, which are

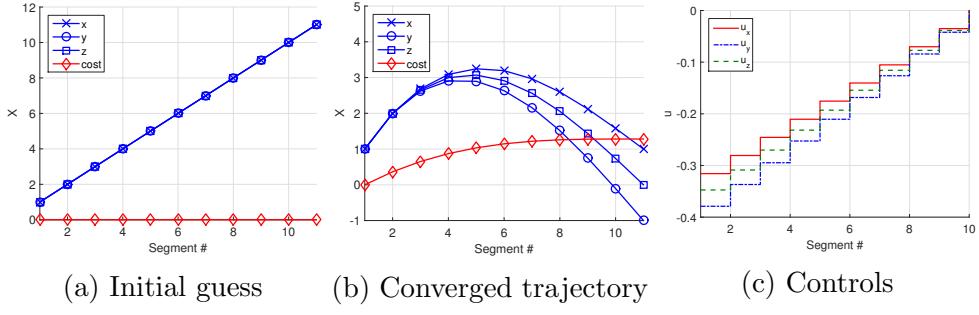


Figure 4.1: Solution of the quadratic linear problem. $J_1 = 1.2807$

solved exactly. An optimal control problem with quadratic objective function subject to linear constraints has a quadratic augmented cost, which will therefore be represented exactly by the expansions in MDDP. The problem is solved in one iteration (when all safeguards are relaxed and the trust-region radii are set to infinity) [265, 179].

The solution obtained with the MDDP algorithm with a single leg is shown in Fig. 4.1. As expected, MDDP converges to the solution in exactly 1 iteration for any number of legs. The expected reductions at each step of the algorithm ($ER_0^{(j)}$, ER_{λ}) match the cost reductions to machine precision, therefore confirming that the quadratic expansions represent the system exactly.

4.2.2 Validation of the Treatment of Terminal Constraints

The example of Section 4.2.1 uses a simple equality terminal constraint: the targeting of a user-specified final state $\mathbf{r}_N = [1, -1, 0]$. The Augmented Lagrangian approach described in Section 3.2 is used to enforce the satisfaction of this linear constraint. However, the same approach can also be used to

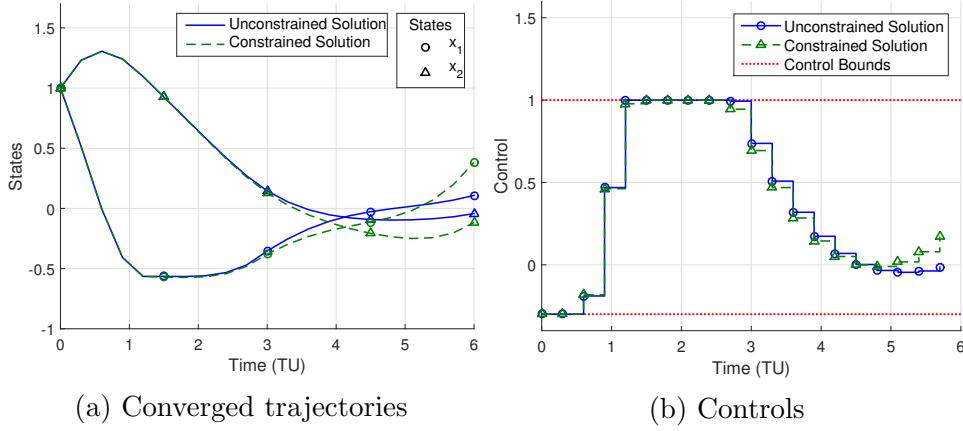


Figure 4.2: Solutions to the constrained and unconstrained Van Der Pol oscillator. $J_1 = 2.8268$ for the unconstrained solution, $J_1 = 2.8658$ for the constrained solution.

enforce more complex terminal constraints, as is demonstrated in this section.

First, the Van Der Pol problem is solved with the terminal constraint described in Eq. (4.11) active and inactive, and $d_f = 0.5$. The solutions presented in Fig. 4.2 demonstrate the respect of the control bounds, as well as the satisfaction of the terminal constraint. As expected, the cost is lower when the problem is unconstrained.

The one-dimensional landing problem of Section 4.1.1.4 is then solved with an inequality constraint on the velocity:

$$v(t_f) \leq 0 \quad \text{or} \quad v(t_f) \geq 0 \quad (4.25)$$

This example is designed to show that inequality constraints are treated like equality constraints when active (in this example, when $v(t_f) \geq 0$), and ignored when inactive. The solution shown in Fig. 4.3 demonstrates the expected bang-bang controls when the constraint is active, and keeps the initial guess of zero

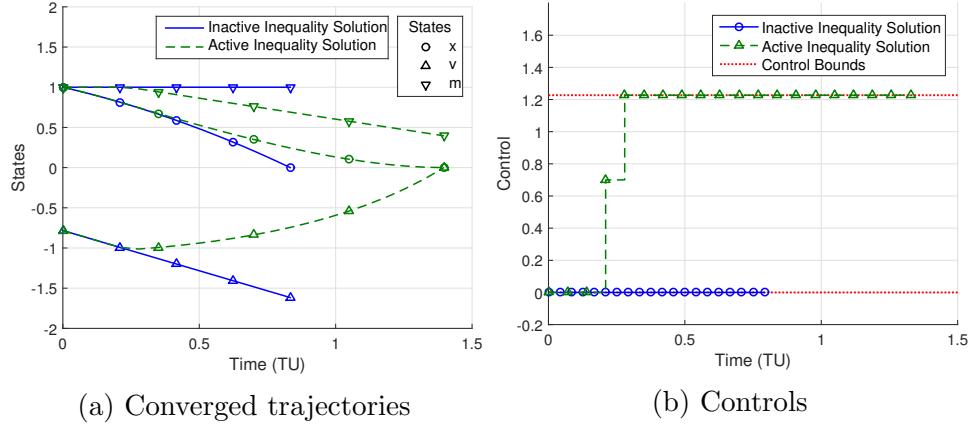


Figure 4.3: Solutions to the 1D landing problem with active and inactive terminal constraint on velocity. $J_1 = -m(t_f) = -1$ MU for the inactive solution, $J_1 = -m(t_f) = -0.39527$ MU for the active solution.

thrust when the constraint is inactive.

Finally, a Hohmann transfer is simulated using the three-dimensional spacecraft of Section 4.1.1.6. The initial conditions are fixed as in Eq. (4.24), the final radius is $r_f = 3$ LU, the control constraints are relaxed, the final time is set to half a revolution of the transfer orbit ($t_f = \pi\sqrt{\frac{2^3}{1}} = 8.8858$ TU), and the problem is solved with active and inactive terminal constraint on circularity ($\mathbf{r}(t_f)^T \mathbf{v}(t_f) = 0$). The trajectories shown in Fig. 4.4 are almost identical to each other, the only difference being the last thrust segment which circularizes the trajectory in the constrained case. As expected, the trajectories are similar to Hohmann transfers, with single thrust segments taking the spacecraft onto the transfer ellipse. The slight rotation of the line of nodes is due to the length of the thrust segment (in contrast to the impulsive ΔV of the Hohmann transfer). The unconstrained solution has a higher final mass, since it does not require a thrust segment to circularize the trajectory.

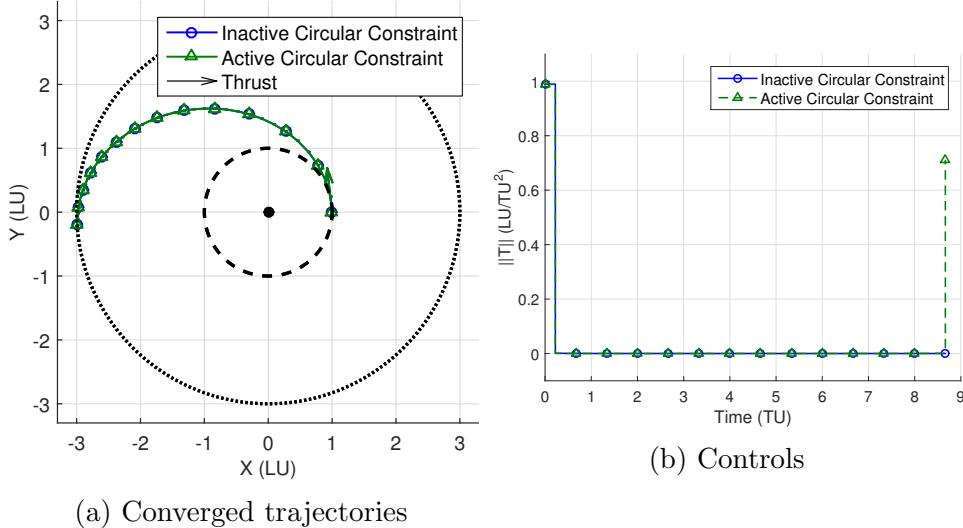


Figure 4.4: Solutions to the 3D Hohmann transfer with active and inactive terminal constraint on circularity. $J_1 = -m(t_f) = -0.95581$ when the circular constraint is inactive, $J_1 = -m(t_f) = -0.92401$ when it is active.

4.2.3 Validation of the Treatment of Path Constraints

In order to confirm that the treatment of the path constraints presented in Section 3.2 is implemented correctly, and yields feasible solutions to the optimal control problems, the problems of Section 4.1.1 are solved with active path constraints. Examples shown in this section use the Brachistochrone problem, the Van Der Pol oscillator, and the 3D spacecraft problem.

The Brachistochrone problem is solved with initial and final conditions $(x_a, y_a) = (0, 0)$ LU and $(x_b, y_b) = (2.0, -2.0)$ LU. The inequality path constraint described by Eq. (4.26) is added to the problem, so that the motion is restricted to be outside of a disk of prescribed center and radius.

$$d_{\min} - [(x - x_c)^2 + (y - y_c)^2] \leq 0 \quad (4.26)$$

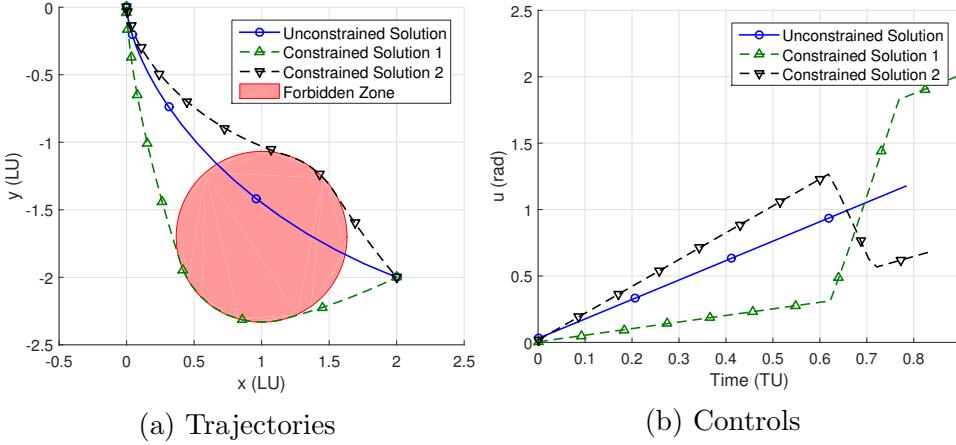


Figure 4.5: Solutions to the constrained and unconstrained Brachistochrone problem. $J_1 = t_f = 0.82446$ TU for the unconstrained solution, $J_1 = t_f = 0.91495$ TU for the constrained solution 1, and $J_1 = t_f = 0.85870$ TU for the constrained solution 2.

with $(x_c, y_c) = (1, -1.7)$ LU and $d_{\min} = 0.4$ LU². The results obtained with the MDDP algorithm are shown in Fig. 4.5. The unconstrained solution, as well as two solutions to the constrained problem obtained with different initial guesses for the controls are exhibited. The adherence to the path constraint is verified.

The Van Der Pol problem with active terminal constraint (see Fig. 4.2) is modified to include the path constraints $(x_1 - x_{1,l} \geq 0)$ and $(x_2 - x_{2,l} \geq 0)$. The bounds are $(x_{1,l}, x_{2,l}) = (-0.4, 0.1)$. The resulting optimal solution is depicted in Fig. 4.6b, which once again demonstrates satisfaction of the path constraints.

Finally, the control constraints of Eq. (4.23) are activated in the Hohmann example of Section 4.2.2. The scalar bounds are $(u_l, u_u) = (10^{-4}, 0.2)$. The vector constraints (bounding each Cartesian element of the thrust using the

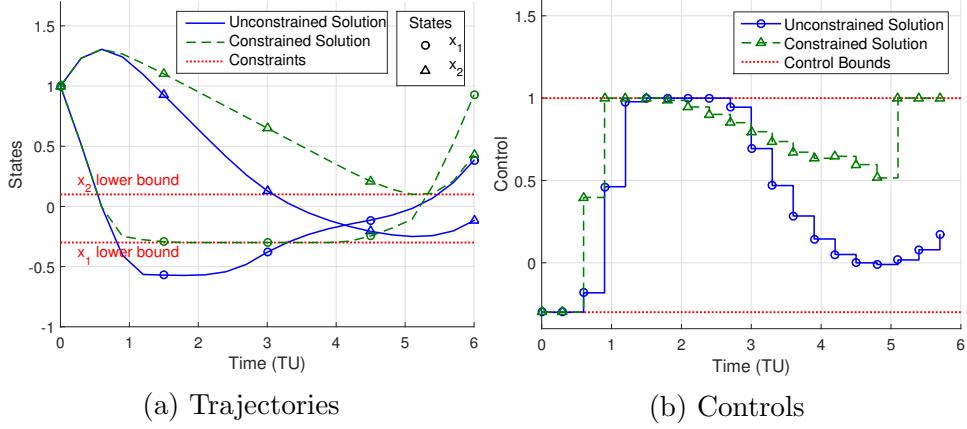


Figure 4.6: Solutions to the constrained and unconstrained Van Der Pol oscillator. $J_1 = 2.8658$ for the unconstrained problem, $J_1 = 4.1716$ for the constrained problem.

null-space trust-region method) are set to be much looser than the thrust magnitude constraint, ensuring that the latter constraint is satisfied using the Augmented Lagrangian approach. The resulting control law, shown in Fig. 4.7, presents the expected bang-bang structure for a maximum-mass (or minimum-fuel) problem.

4.3 Performance Analysis of the Multiple-Shooting Formulation

The single-shooting version of MDDP, including the treatment of equality and inequality path and terminal constraints, has been validated in Section 4.2.1. The present section displays applications of the multiple-shooting formulation to different optimal control problems. First, the expansions of Section 3.3 are verified once more using the quadratic linear problem and 10 legs. The solution of Fig. 4.8 is obtained in exactly one iteration, and all ex-

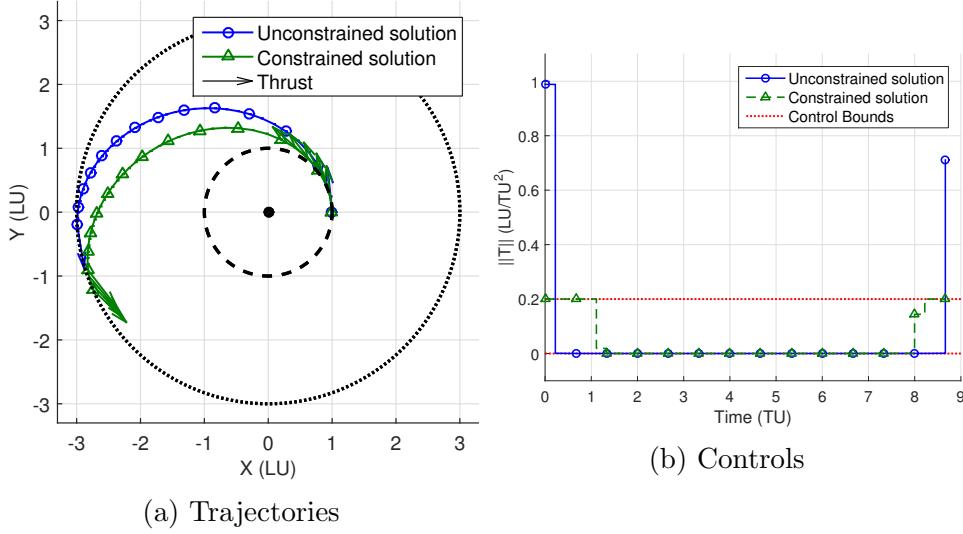


Figure 4.7: Solutions to the 3D Hohmann transfer with inactive and active path constraint. The final cost is $J_1 = -0.92401$ for the unconstrained solution, and $J_1 = -0.90887$ for the constrained problem.

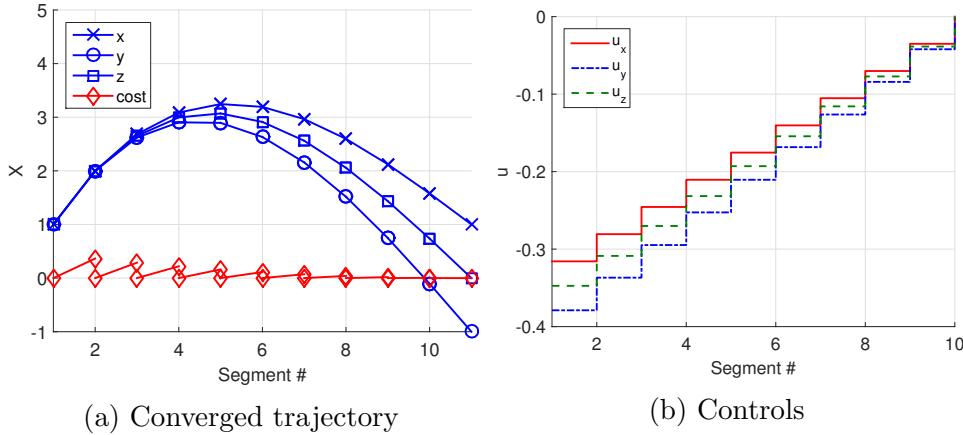


Figure 4.8: Solutions of the quadratic linear problem with 10 legs.
 $J = 1.2807$

pected reductions match real cost reductions to machine precision. Note that the cost is discontinuous, since it is equivalent to a cost of Mayer, and can be set to 0 at the beginning of each leg, as explained in Eq. (3.13).

The set of test cases used to demonstrate the functionality of the multiple-shooting formulation is described hereafter:

- **Brachistochrone:** The variations of the Brachistochrone problem presented in [Section 4.2.1](#) are denoted by `Brach1` for the unconstrained problem and `Brach2` for the constrained problem. Their solutions are given in [Fig. 4.5](#).
- **Van Der Pol oscillator:** `VDP1` and `VDP2` designate the unconstrained and constrained Van Der Pol problems of [Section 4.2.3](#), respectively. Their solutions are given in [Fig. 4.6](#).
- **Minimum-time Van Der Pol:** The problems noted `VDP3` and `VDP4` use the $J_2 = t_f$ cost function of [Eq. \(4.9\)](#). This variant is a minimum-time problem with linear controls, and is expected to have a bang-bang structure, which is exhibited in [Figure 4.9b](#). The problem is solved with inactive (`VDP3`) and active (`VDP4`) path constraints, and the solutions obtained with 20 legs for both problems are shown in [Fig. 4.9](#).
- **1D landing:** The one-dimensional landing problem of [Section 4.2.2](#) is noted `1D`, and its solution is presented in [Fig. 4.3](#).
- **2D Spacecraft:** The two-dimensional spacecraft problem in polar coordinates is used to simulate a planar orbital transfer from a circular orbit at radius 1 LU to a circular orbit at 3 LU, as in the 3D spacecraft problem. The thrust is bounded $[(u_l, u_u) = (0.01, 0.2)]$ using the

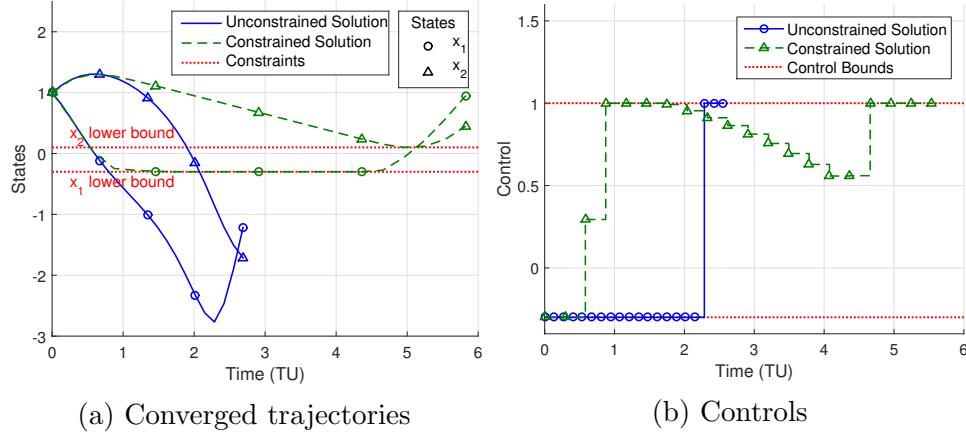


Figure 4.9: Solutions of the path constrained and unconstrained minimum-time Van Der Pol oscillator. $J_2 = t_f = 2.6889$ TU for the unconstrained solution, and $J_2 = t_f = 5.8242$ TU for the constrained problem.

null-space trust-region method, or using the Augmented Lagrangian approach, which results in identical converged solutions, but different iteration counts. 2DS1 utilizes J_2 and the null-space trust-region method, 2DS2 uses J_2 and the Augmented Lagrangian approach, while 2DS3 and 2DS4 optimize J_1 and use the null-space and Augmented Lagrangian approaches respectively. The solution for both the minimum-fuel (J_1) and the minimum quadratic thrust (J_2) are shown in Fig. 4.10.

- **3D Spacecraft:** The three-dimensional spacecraft problem introduced in Section 4.2.3, with active path constraints, is noted 3DS2, and its solution is in Fig. 4.7. 3DS1 designates the minimum quadratic thrust (using J_2) variant of the same problem.

The number of iterations required to obtain convergence on each of these 13 problems is shown in Table 4.2, with varying number of legs $n_l \in$

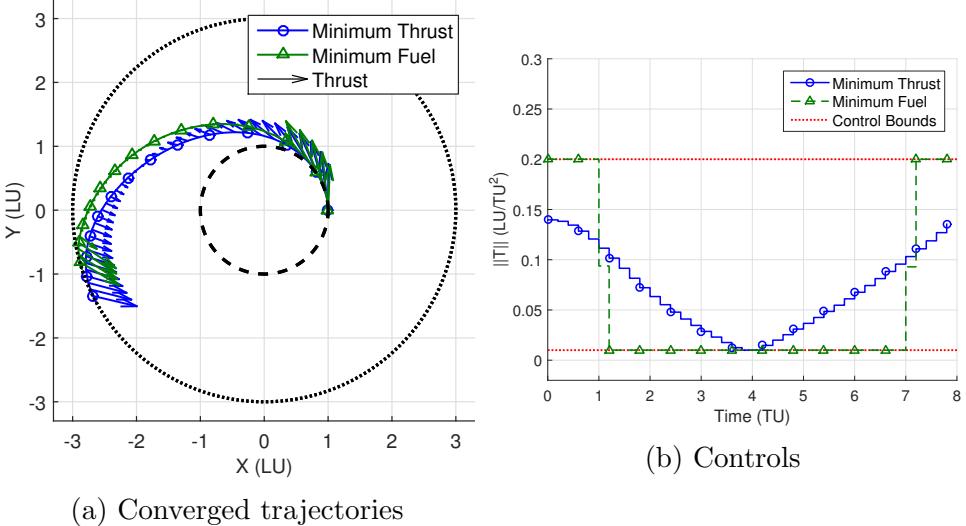


Figure 4.10: Solutions of the minimum-fuel (2DS3 and 2DS4) and minimum-thrust (2DS1 and 2DS2) 2D spacecraft problem.

$J_1 = -m(t_f) = -0.90893$ for the minimum-fuel problem, and $J_2 = 2.6449$ for the minimum thrust problem.

[1, 2, 4, 5, 10, 20], resulting in 65 multiple-shooting test cases. The multiple-shooting formulation fails to converge in a single case, for 3DS1 with 10 legs. While this failure could be remedied by altering the initial guess or some of the algorithm parameters, the present section aims at giving statistical insights for the baseline settings of Section 4.1.2, rather than finely tune each example. The converged solutions for multiple-legs are identical to the single-shooting version, up to the prescribed convergence tolerance. Table 4.2 demonstrates the capability of the MDDP algorithm to solve optimal control problems using a multiple-shooting formulation.

Table 4.2 also shows that, while the multiple-shooting formulation converges in most cases, it does not usually lower the iteration count or improve the convergence rate. The examples used in this section have a relatively low

Table 4.2: Number of iterations for the collection of problems

Problem \ # of legs	1	2	4	5	10	20
Brach1	21	61	137	75	98	125
Brach2	28	190	180	84	103	123
VDP1	22	29	36	38	42	44
VDP2	56	70	78	67	71	81
VDP3	147	81	110	106	102	293
VDP4	76	91	86	64	87	86
1D	48	51	75	75	132	273
2DS1	46	50	52	72	93	84
2DS2	73	114	103	77	131	158
2DS3	104	176	386	338	162	322
2DS4	167	168	324	260	282	353
3DS1	51	104	70	94	N/A	754
3DS2	65	179	336	396	273	1948

complexity, especially when compared to multiple-body spacecraft trajectories. Therefore, the benefits of splitting the sensitivities are not as important as the increase in complexity due to the addition of unknown states and of linkage constraints. The results obtained with a single-leg on the two-dimensional spacecraft problem also demonstrate that, when simple bounds are present, it is more efficient to treat them using the null-space trust-region method (2DS1 and 2DS3) than using the Augmented Lagrangian approach (2DS2 and 2DS4), as suggested in Ref. [72].

4.3.1 Multiple-Revolution Orbit transfer

In this example, which uses the two-dimensional spacecraft problem, the time of flight is chosen to yield approximately 40 revolutions for an orbital transfer from a circular orbit of radius 1 LU to a circular orbit of radius 3 LU. The objective function is chosen to be a quadratic function of the thrust, yielding a smoother solution than the typical minimum-fuel problem. The solution is presented in Fig. 4.11, and the controls display the expected sinusoidal shape (small variations about the velocity direction). The long time of flight and multiple revolutions yield a highly sensitive trajectory, a difficulty that can be mitigated by the multiple-shooting formulation. In order to demonstrate this capability, the norm of the displacement predicted by the first-and second-order STM is plotted as a function of the segment number in Fig. 4.12. This proxy metric for sensitivity is calculated as:

$$\Delta_k = \frac{\|\delta\mathbf{X}_k\|}{\|\delta\mathbf{X}_0\|} = \frac{\Phi^1(k, 0)\delta\mathbf{X}_0 + \frac{1}{2}\delta\mathbf{X}_0^T \bullet_2 \Phi^2(k, 0)\delta\mathbf{X}_0}{\|\delta\mathbf{X}_0\|} \quad (4.27)$$

where, if \mathbf{v} is an $N \times 1$ vector and T an $N \times N \times N$ tensor:

$$(\mathbf{v} \bullet_2 T)(i, j) = \sum_{p=1}^N T(i, p, j)v(p) \quad (4.28)$$

The sensitivity of the final state with respect to initial conditions is significantly decreased when using multiple legs, as demonstrated in Fig. 4.12. This well-known property of multiple-shooting methods increases their robustness to bad initial guesses. This property can enable optimal control solutions to highly sensitive problems that are otherwise intractable using single shooting methods. Figure 4.12 shows the sensitivities are reduced by about two

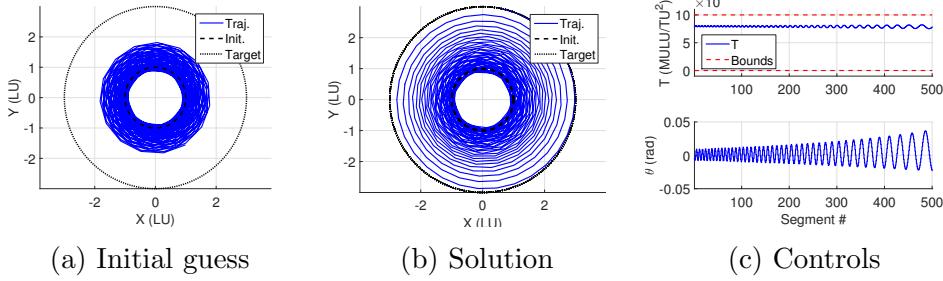


Figure 4.11: Solution of the orbital transfer problem

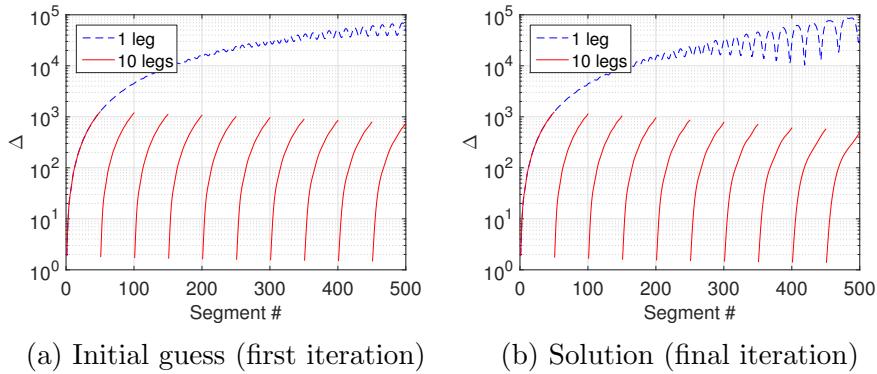


Figure 4.12: Sensitivity metric as computed in Eq. (4.27)

orders of magnitude when using 10 legs instead of 1. Note that in this case, the displacements Δ_k for the first iteration or the solution are quite similar, which is understandable since the initial trajectory is relatively close to the solution.

4.3.2 Quasi-Newton Numerical Results

In this section, the quasi-Newton modification of the MDDP algorithm (QMDDP) is tested on the optimal control problems presented in [Section 4.3](#), in order to analyze the performance of the quasi-Newton approximations to the second-order STM. Several scenarios are examined, and compared to the full

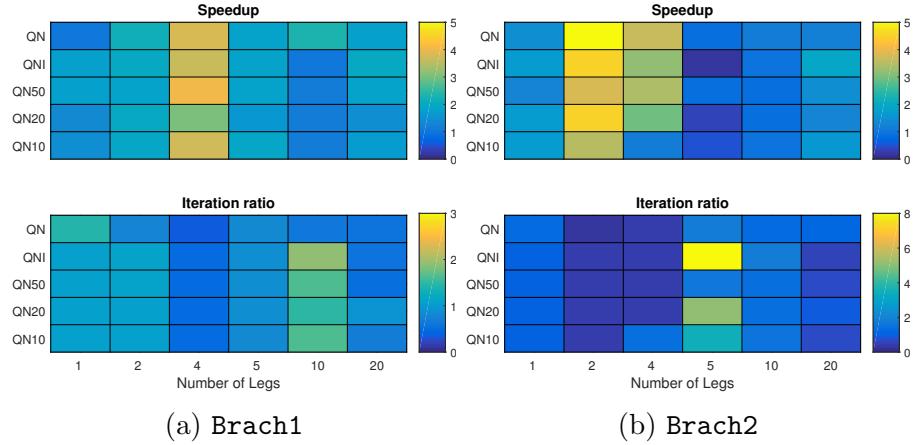


Figure 4.13: Speedup and iteration ratio of the quasi-Newton scenarios for the Brachistochrone problems

second-order algorithm (noted SO). A pure quasi-Newton method, denoted QN, does not utilize any second-order information and relies on approximations only. The second-order STM is initialized at 0, and subsequent iterations improve the approximation. In the QNI scenario, the full second-order STM is evaluated at the first iteration, in order to give a better initial guess to the quasi-Newton method. In the QN10, QN20, and QN50 scenarios, the full-second order STM is evaluated every 10, 20 and 50 iterations respectively. Figures 4.13 to 4.17 contain the speedups [Eq. (4.29)] and iteration ratio [Eq. (4.30)] for all 13 problems, varying the number of legs, and quasi-Newton scenario. A barred white cell indicates that the corresponding problem did not converge.

$$SU = \frac{t_{\text{tot,SO}}}{t_{\text{tot,QN}^*}} \quad (4.29)$$

$$It = \frac{n_{\text{iter},QN^*}}{n_{\text{iter},SO}} \quad (4.30)$$

As expected, the algorithm is usually faster when using quasi-Newton

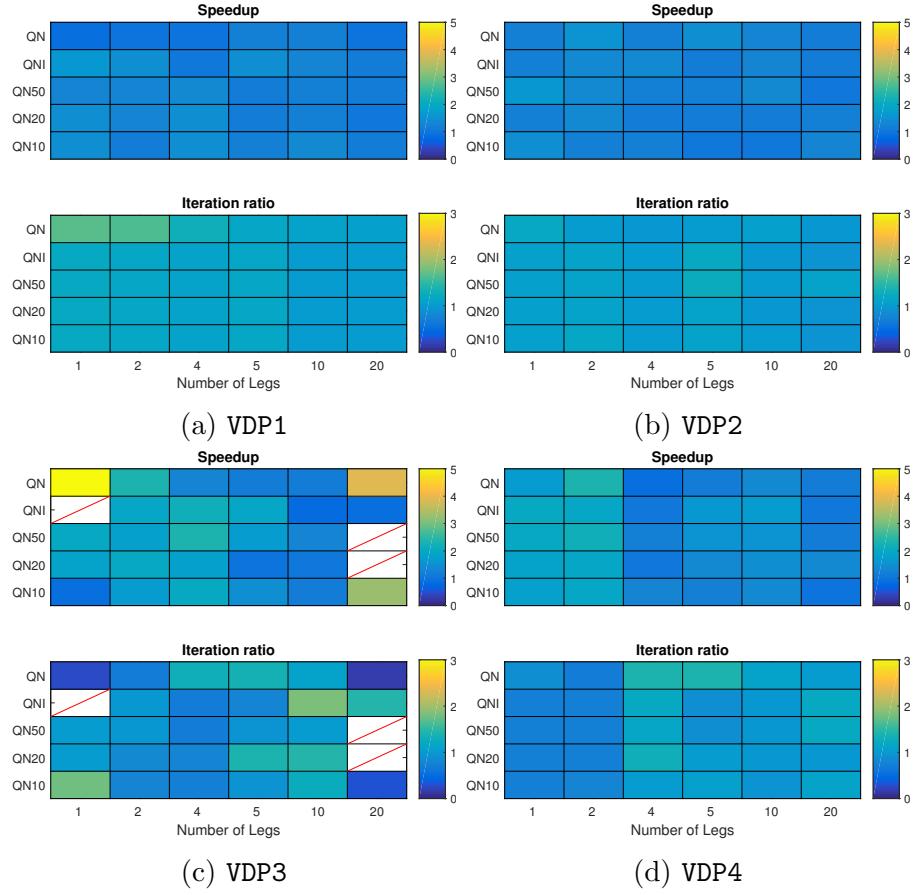


Figure 4.14: Speedup and iteration ratio of the quasi-Newton scenarios for the Van Der Pol problems

approximations, yet requires more iterations to reach convergence. The increase in iteration number is due in part to the inaccurate second-order information yielding imperfect descent directions. The QMDDP algorithm also tends to take smaller steps: the computation of the expected reduction uses the second-order sensitivities, and is corrupted by the quasi-Newton approximations. The trust-region management strategy explained in [Section 3.4.2](#) therefore yields reduced trust-region radii as compared to the full second-order

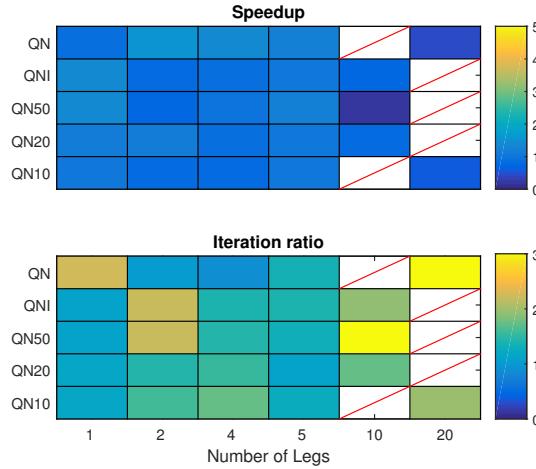


Figure 4.15: Speedup and iteration ratio of the quasi-Newton scenarios for the one-dimensional landing problem

case.

In spite of the increased iteration count, the quasi-Newton approach yields lower runtimes for most application cases. Indeed, the most computationally intense step of the MDDP algorithm is the evaluation of the full second-order STM. The maximum speedup per iteration can be computed for each test case:

$$SU_{\max} = \frac{t_{\text{iter}}}{t_{\text{iter}} - t_{\text{SO}} + t_{\text{FO}}} \quad (4.31)$$

where t_{iter} (s) is the average runtime per iteration, t_{SO} (s) is the time required for a full second-order STM computation, and t_{FO} (s) is the time required for a first-order STM computation, which is still necessary in the quasi-Newton scenarios. [Figure 4.18](#) shows the maximum speedups per iteration for each test case. As expected, the speedups are tied to the complexity of the equations of motion, suggesting that the quasi-Newton approach will be more beneficial for complex dynamics.

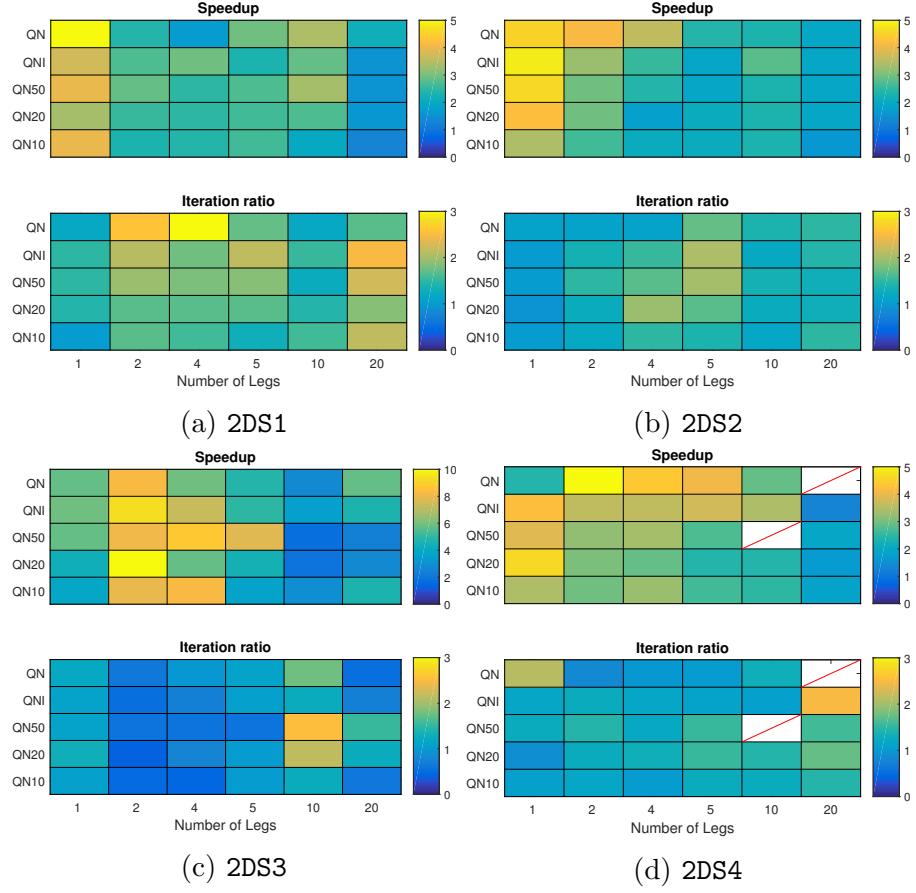


Figure 4.16: Speedup and iteration ratio of the quasi-Newton scenarios for the two-dimensional spacecraft problems

The results of Figs. 4.13 to 4.17 show that the maximum speedup is rarely attained, due to the higher iteration count. The impact of the quasi-Newton approximations is also decreased when the number of legs is increased. This is explained by the additional computations necessary for the initial conditions and Lagrange multipliers updates when adding unknown states and new linkage conditions. The computation of the full second-order STM becomes less prevalent on the total runtime when adding legs. Moreover, it

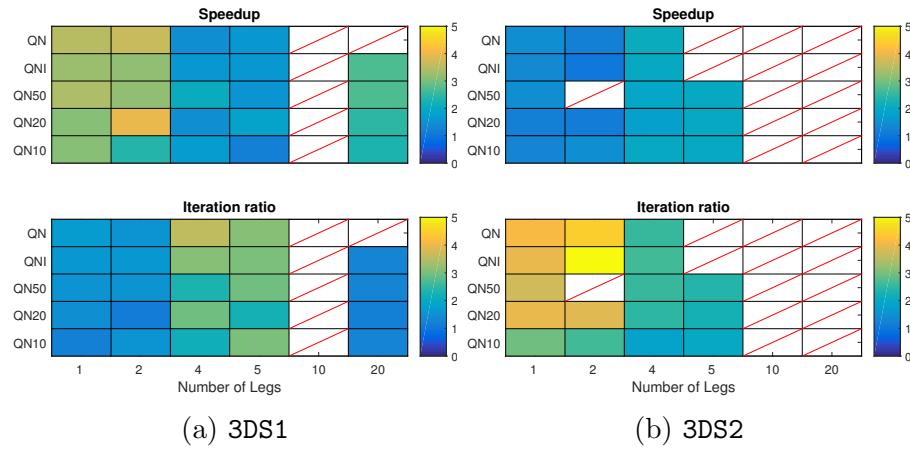


Figure 4.17: Speedup and iteration ratio of the quasi-Newton scenarios for the three-dimensional spacecraft problems

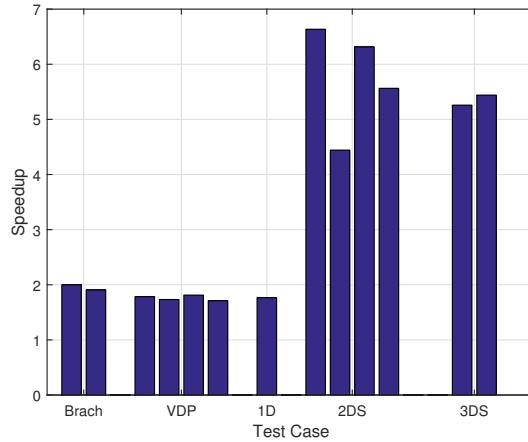


Figure 4.18: Maximum speedup per iteration

is apparent that the different quasi-Newton scenarios yield similar speedups. Therefore, utilizing an initialization strategy such as QN10 when the second-order partials are available allows to reduce the runtime while limiting the decrease in robustness, since the full second-order STM is computed every 10 iterations.

For completeness, other quasi-Newton updates were tested in the pre-

sented framework: the DFP update, the BFGS update, as well as Powell’s Damped Algorithm. Consistent with the discussion in [Section 3.5.2](#), none of these convexity preserving updates originated acceptable results, as the algorithm did not converge for most of the benchmark problems. In addition, a simple first-order method was tested: the second-order STM was set to $0_{N \times N \times N}$ for each iteration, therefore canceling any feedback controls and second-order effects. Similarly, the resulting algorithm did not converge for most benchmark problems.

Chapter 5

On the Computation and Accuracy of Trajectory State Transition Matrices

In this chapter¹, the different methods for partials computation presented in [Section 1.2.2](#) are described, and compared in terms of timing and accuracy. The differences between using a variable- or a fixed-step integrator for the computations are emphasized. A number of problems appear when using variable-step integrators, because of the changing integration path. These subtleties are examined in detail and recommendations are made to improve the quality of the partials obtained by practitioners. The first part of the chapter provides explanations of the three methods used for sensitivity computations, as well as some details of their implementation. The results obtained for different tests and metrics on a variety of orbital problems are presented in the second part.

¹Work from this chapter was published in a peer-reviewed journal as:

- Etienne Pellegrini and Ryan P. Russell, On the Computation and Accuracy of Trajectory State Transition Matrices, *Journal of Guidance, Control, and Dynamics*, Vol. 39 No.11, pp.2485-2499, November 2016, [doi:10.2514/1.G001920](https://doi.org/10.2514/1.G001920)

Work from this chapter was presented as:

- **Etienne Pellegrini** and Ryan P. Russell, On the Accuracy of Trajectory State-Transition Matrices, Paper AAS 15-785, *AAS/AIAA Astrodynamics Specialists Conference*, Vail, CO, August 2015

5.1 Methods

This section presents three common methods used for the computation of partial derivatives, which were introduced in [Section 1.2.2](#). The methods are applied to the computation of the first- and second-order STMs (see [Section 2.1.1](#)) which are essential to the MDDP algorithm.

5.1.1 Variational Equations

The variational equations, when available, allow the practitioner to propagate the STMs directly alongside the trajectory. The equations are added to the equations of motion (EOMs), and the state is augmented with the first- and second-order STMs. Consider a state vector containing N elements, and subject to the EOMs:

$$\dot{\mathbf{X}} = \mathbf{f}(\mathbf{X}) \quad (5.1)$$

The dynamics function \mathbf{f} is assumed to be well-behaved (at least of class C^2 over the whole domain). If time explicitly appears in the right hand side of [Eq. \(5.1\)](#) it can be removed by augmenting the state with a time element. Then the variational equations are:

$$\dot{\Phi}^1 = f_X \Phi^1 \quad (5.2)$$

$$\dot{\Phi}^2 = f_X \bullet_1 \Phi^2 + \Phi^{1T} \bullet f_{XX} \bullet \Phi^1 \quad (5.3)$$

where f_X is the Jacobian matrix of the dynamics function f , f_{XX} is its Hessian, and the \bullet operator is defined in [Section 2.1](#).

Propagating the STMs alongside the state means propagating $N+N^2+N^3$ equations simultaneously, which can have an impact on the integration

path (see [Section 5.2.2.1](#)). It is possible to take advantage of the symmetry of the second-order partials [[Eq. \(2.16\)](#)], but this results in a loss of the matrix notation, and the resulting expressions are tedious to implement with the variational equations, since [Eq. \(5.2\)](#) and [Eq. \(5.3\)](#) use matrix algebra. In this study, symmetry is not considered with the variational equations. However, it is noted that judicious code optimization will identify repeated expressions, thus accounting to first order for any symmetry automatically. The size of the state, however, remains $N + N^2 + N^3$.

In order to propagate the STMs, the f_X matrix and f_{XX} tensor must be available. Symbolic manipulation software (Maple) is used to obtain analytical derivatives of the dynamics. The required partials are computed analytically, and Maple's code generation capabilities are used to create optimized Fortran code for each case (f , $\{f, f_X\}$, $\{f, f_X, f_{XX}\}$). Symbolic manipulation software can often yield inefficient code, due to insufficient factoring, generating repeated computations. Maple offers an `optimize` option, which helps reducing repeat computations. An alternative but less efficient way of obtaining the variational equations would be to use the finite difference or CSD methods on the dynamics function alone.

When using numerical integration to propagate equations of motion and variational equations, the resulting state trajectory is an approximation of the true trajectory up to an order that depends on the integrator. However, the computed STMs are the derivatives of the approximated trajectory (to roundoff error) for the fixed-step integrator. In the simplest case of the Euler integrator with a fixed-step δt , the update equation for the state and first-order

STM are:

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \mathbf{f}(\mathbf{X}_i)\delta t \quad (5.4)$$

$$\Phi_{i+1}^1 = \Phi_i^1 + f_X|_{\mathbf{X}_i} \Phi_i^1 \delta t \quad (5.5)$$

Then, by definition of Φ^1 :

$$\Phi_{i+1}^1 = \Phi^1(t_{i+1}, t_0) = \frac{\partial \mathbf{X}_{i+1}}{\partial \mathbf{X}_0} \quad (5.6)$$

Taking the partial of Eq. (5.4) with respect to X_0 yields:

$$\frac{\partial \mathbf{X}_{i+1}}{\partial \mathbf{X}_0} = \frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_0} + \left. \frac{\partial \mathbf{f}}{\partial \mathbf{X}} \right|_{\mathbf{X}_i} \frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_0} \delta t \quad (5.7)$$

$$= \Phi_i^1 + f_X|_{\mathbf{X}_i} \Phi_i^1 \delta t \quad (5.8)$$

Equation (5.5) and Eq. (5.8) are identical, demonstrating that the approximation of the STM generated by the integrator [Eq. (5.5)] is indeed the partial of the approximated state generated by the same integrator on the same path [Eq. (5.8)].

In the case of a variable-step integrator, the time increment δt is computed from the approximation of the error at the previous step ($\delta t = \delta t(\mathbf{X}_i)$) [267]. Therefore, Eq. (5.8) becomes:

$$\frac{\partial \mathbf{X}_{i+1}}{\partial \mathbf{X}_0} = \frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_0} + \left. \frac{\partial \mathbf{f}}{\partial \mathbf{X}} \right|_{\mathbf{X}_i} \frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_0} \delta t + \mathbf{f}(\mathbf{X}_i) \left. \frac{\partial \delta t}{\partial \mathbf{X}} \right|_{\mathbf{X}_i} \frac{\partial \mathbf{X}_i}{\partial \mathbf{X}_0} \quad (5.9)$$

$$= \Phi_i^1 + \left(f_X|_{\mathbf{X}_i} \delta t + \mathbf{f}(\mathbf{X}_i) \left. \frac{\partial \delta t}{\partial \mathbf{X}} \right|_{\mathbf{X}_i} \right) \Phi_i^1 \quad (5.10)$$

The $\frac{\partial \delta t}{\partial \mathbf{X}}$ term in Eq. (5.10) is not found in Eq. (5.5). Therefore, the variational equations do not capture the changes to the state due to a change in the

integration step sizes. Similar derivations can be made for the second-order STM, and for higher order integrators. The partial derivative of the integration step size was also derived by Ellison et al. [95], in order to compute time of flight derivatives. This subtle shortfall of the variational equations when propagated with variable-step integrators will be demonstrated in [Section 5.2.2.2](#). The result, however, is consistent with the practitioners' experience that fixed-step integrators often work better than their variable-step counterparts when embedded inside a root-solving or optimization algorithm [25].

For the variable-step integrator used in this chapter (Runge-Kutta-Fehlberg (7)8), the δt update equation is:

$$\delta t_{i+1} = \delta t(\mathbf{X}_i) = 0.8\delta t_i \left(\frac{\epsilon u(\mathbf{X}_i)}{v(\mathbf{X}_i)} \right)^{\frac{4}{5}} \quad (5.11)$$

where ϵ is the tolerance input to the variable-step integrator. The function $u(\mathbf{X}_i) = \|\mathbf{X}_i\|$ is used to scale the tolerance to the considered problem. The function $v(\mathbf{X}_i)$ is the embedded error, and is computed from the difference between the approximations of order 7 and 8. Then:

$$\frac{\partial \delta t}{\partial \mathbf{X}} \Big|_{\mathbf{X}_i} = \epsilon^{\frac{4}{5}} 0.64\delta t_i \left(\frac{u(\mathbf{X}_i)}{v(\mathbf{X}_i)} \right)^{-\frac{1}{5}} \left(\frac{\frac{\partial u}{\partial \mathbf{X}}|_{\mathbf{X}_i} v(\mathbf{X}_i) - u(\mathbf{X}_i) \frac{\partial v}{\partial \mathbf{X}}|_{\mathbf{X}_i}}{v(\mathbf{X}_i)^2} \right) \quad (5.12)$$

The $\frac{\partial \delta t}{\partial \mathbf{X}}$ term in [Eq. \(5.10\)](#) is a factor of $\epsilon^{\frac{4}{5}}$, and will therefore have a greater impact for low accuracy propagations, when the tolerance is loose, as is demonstrated in [Section 5.2.2.2](#).

5.1.2 Complex Step Derivative

The complex step approach is based on the Taylor Series of complex analytic functions, and is described extensively in [202]. The resulting expression

for the first-order derivative of an analytic function f is:

$$f'(x) = \frac{\Im(f(x + ih))}{h} + \mathcal{O}(h^2) \quad (5.13)$$

The main difference between this complex approach and the classic finite differences methods (see [Section 5.1.3](#)) is the absence of subtraction between two function evaluations. This absence allows the perturbation h to have a very small magnitude, as it will not introduce subtraction cancellation error. Therefore the truncation error can be made arbitrarily small, and the derivative can be approximated to the same precision as the function itself. Martins demonstrated that the CSD approximation is similar to the forward mode of Algorithmic Differentiation methods, with some extra computations being performed for the CSD method [\[201\]](#).

The CSD approximation was extended to high-order derivatives by Lan-toine et al. [\[181\]](#) with the use of multi-complex numbers, a multi-dimensional generalization of complex numbers. In order to obtain the second-order STMs studied in this chapter, bi-complex numbers are necessary. Bi-complex numbers $q \in \mathbb{C}^2$ are defined as:

$$\mathbb{C}^2 = \{z_1 + i_2 z_2 \mid z_1, z_2 \in \mathbb{C}\} = \{x_1 + i_1 x_2 + i_2 x_3 + i_1 i_2 x_4 \mid x_1, x_2, x_3, x_4 \in \mathbb{R}\} \quad (5.14)$$

Then the first- and second-order derivatives of a function of two variables are

computed as such:

$$\frac{\partial^2 f(x, y)}{\partial x^2} = \frac{\Im_{12}(f(x + hi_1 + hi_2, y))}{h^2} \quad (5.15)$$

$$\frac{\partial^2 f(x, y)}{\partial y^2} = \frac{\Im_{12}(f(x, y + hi_1 + hi_2))}{h^2} \quad (5.16)$$

$$\frac{\partial^2 f(x, y)}{\partial x \partial y} = \frac{\Im_{12}(f(x + hi_1, y + hi_2))}{h^2} \quad (5.17)$$

$$\frac{\partial f(x, y)}{\partial x} = \frac{\Im_1(f(x + hi_1 + hi_2, y))}{h} = \frac{\Im_2(f(x + hi_1 + hi_2, y))}{h} \quad (5.18)$$

$$\frac{\partial f(x, y)}{\partial y} = \frac{\Im_1(f(x, y + hi_1 + hi_2))}{h} = \frac{\Im_2(f(x, y + hi_1 + hi_2))}{h} \quad (5.19)$$

where, if $q = x_1 + i_1x_2 + i_2x_3 + i_1i_2x_4$,

$$\Im_1(q) = x_2 \quad \Im_2(q) = x_3 \quad \Im_{12}(q) = x_4 \quad (5.20)$$

The implementation of the CSD and multi-complex approaches are described by Martins et al. [202] and Lantoine et al. [181] respectively. A Fortran dynamic library implemented by the author, which includes all the operator and intrinsic functions overloading needed to work with complex and bi-complex numbers, is used in the present study. The code for the real number implementation of the function requiring differentiation needs to be modified so that it can accept complex and bi-complex arguments. In the case of trajectory propagation, the function itself is the whole integration, and the variable definitions parts of the code for the integrators and dynamics have to be modified. A Python preprocessor called **genearize** is written to execute those changes. It takes Fortran files as an input and returns a “generic” version of the files: each procedure contained in the files is duplicated, the type of the variables is changed, and an interface is written, allowing all procedures

in the file to be called with real, complex or bi-complex arguments. With the CSD algorithm, it is trivial to take advantage of the symmetry of the second-order STM [Eq. (2.16)]. Accordingly, the implementation used in this chapter, as well as in the rest of the dissertation, computes only the necessary unique terms. Comparisons of the computational times needed for the integration of complex or bicomplex states to the other approaches are made in [Section 5.2.3](#).

The CSD method propagates the EOMs using complex or bi-complex numbers, whereas the state is usually propagated using only double precision real numbers. Therefore, the roundoff error in the dynamics computations can be different when propagating the STMs or the state. This minor difference in roundoff error creates a difference in the error estimates, which is amplified when those estimates are used to compute the next step size. Changing the integration path yields inaccuracies in the partials. However, the differences are negligible for the cases considered in the current chapter. While CSD and AD have been proved to be linked [202], automatic differentiation should not present the same behavior, as it does not use complex numbers for the propagation.

When using the CSD approximation, the whole integration process is computed using complex or bi-complex numbers. Therefore, the CSD generates approximations of the derivatives of the whole process, including the embedded step size adjustment. Including the sensitivity of the step size term is a subtle but important advantage of CSD and AD methods over the variational equations, when using a variable-step integrator (see [Section 5.2.2.2](#)).

5.1.3 Finite Differences

A common technique for numerical differentiation is the finite difference method, because it does not require extra dynamics (as for the variational equations), and is relatively easy to implement. Moreover, using the variational equations or the CSD method is not always possible, for instance when using black-box codes, such as linear algebra packages. The different formulas for finite differences are obtained from the truncation and combination of Taylor series expansions. The simplest examples of such methods are the forward differences (FD) and central differences (CD), computed as such for a function of one variable:

$$f'(x)_{\text{FD}} = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h) \quad (5.21)$$

$$f'(x)_{\text{CD}} = \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2) \quad (5.22)$$

5.1.3.1 Generic Finite Differences

The expressions in Eq. (5.21) and Eq. (5.22) suffer from a truncation error depending on the magnitude of the perturbation h . However, they also suffer from floating point arithmetic errors, especially when h becomes small [156, 115]. Therefore, a generic finite differences algorithm is implemented, which computes the stencil and coefficients needed for any order approximation of the derivatives. Higher-order approximations yield smaller truncation errors, and the magnitude of h can be increased, therefore reducing the floating point arithmetic error as well. The generic finite differences approximation

[116] is written:

$$\frac{h^d}{d!} f^{(d)}(x) = \sum_{i=1}^{n+1} \beta_i f(x + \alpha_i h) + \mathcal{O}(h^{n+1}) \quad (5.23)$$

$$f^{(d)}(x) = \frac{d!}{h^d} \sum_{i=1}^{n+1} \beta_i f(x + \alpha_i h) + \mathcal{O}(h^{n+1-d}) \quad (5.24)$$

where α_i 's comprise the stencil, and β_i 's are the coefficients. For first-order partials:

$$f'(x) = \frac{1}{h} \sum_{i=1}^{n+1} \beta_i f(x + \alpha_i h) + \mathcal{O}(h^n) \quad (5.25)$$

The forward differences formula of Eq. (5.21) uses $n = 1$, $\alpha = [0, 1]$ and $\beta = [-1, +1]$. The central differences formula of Eq. (5.22) has $n = 2$ and $\alpha = [-1, 0, 1]$ and $\beta = [-\frac{1}{2}, 0, +\frac{1}{2}]$. In general, for an arbitrary order n , the Taylor series expansions can be expressed as:

$$f(x + \alpha h) = \sum_{j=0}^n \frac{\alpha^j h^j}{j!} f^{(j)}(x) + \mathcal{O}(h^n) \quad (5.26)$$

Then Eq. (5.25) can be written:

$$f'(x) = \frac{1}{h} \sum_{i=1}^{n+1} \beta_i \sum_{j=0}^n \frac{\alpha_i^j h^j}{j!} f^{(j)}(x) + \mathcal{O}(h^n) \quad (5.27)$$

$$f'(x) = \frac{1}{h} \sum_{i=1}^{n+1} \sum_{j=0}^n \beta_i \alpha_i^j \frac{h^j}{j!} f^{(j)}(x) + \mathcal{O}(h^n) \quad (5.28)$$

Equation (5.28) equation can be expressed in matrix form:

$$\begin{aligned}
f'(x) &= \frac{1}{h} \left(\begin{bmatrix} 1 & \dots & 1 \\ \alpha_1 h & \dots & \alpha_{n+1} h \\ \frac{1}{2} \alpha_1^2 h^2 & \dots & \frac{1}{2} \alpha_{n+1}^2 h^2 \\ \dots & \dots & \dots \\ \frac{1}{n!} \alpha_1^n h^n & \dots & \frac{1}{n!} \alpha_{n+1}^n h^n \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_{n+1} \end{bmatrix} \right)^T \begin{bmatrix} f(x) \\ f'(x) \\ f''(x) \\ \vdots \\ f^{(n)}(x) \end{bmatrix} \\
&= \frac{1}{h} \left(\begin{bmatrix} 1 & \dots & 1 \\ \alpha_1 h & \dots & \alpha_{n+1} h \\ \alpha_1^2 h^2 & \dots & \alpha_{n+1}^2 h^2 \\ \dots & \dots & \dots \\ \alpha_1^n h^n & \dots & \alpha_{n+1}^n h^n \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_{n+1} \end{bmatrix} \right)^T \begin{bmatrix} f(x) \\ f'(x) \\ \frac{1}{2} f''(x) \\ \vdots \\ \frac{1}{n!} f^{(n)}(x) \end{bmatrix} \quad (5.29)
\end{aligned}$$

In order for Eq. (5.29) to be verified, the coefficients α_i and β_i must be a solution of:

$$\begin{bmatrix} 1 & \dots & 1 \\ \alpha_1 & \dots & \alpha_{n+1} \\ \alpha_1^2 & \dots & \alpha_{n+1}^2 \\ \dots & \dots & \dots \\ \alpha_1^n & \dots & \alpha_{n+1}^n \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_{n+1} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (5.30)$$

The system in Eq. (5.30) is called a Vandermonde system, and efficient algorithms are available for solving it [39]. The implementation used in the present work is based on the solution found in Ref. [267]. The stencil is taken to be equally spaced, and constructed to correspond to either forward, backward or central differences. The resulting coefficients have been checked using the known solutions found by Fornberg [116].

High-order approximations of the derivatives usually improve the accuracy, but they also present drawbacks: 1) the high number of function evaluations obviously increases the compute time, although the trade-off may be worth it for very sensitive problems where the accuracy of the partials is essential to convergence; 2) the function must remain continuous for all the

evaluations (meaning for all α_i 's), which widens the range around the reference point where the function must be continuous. In the particular case of STM computations using a variable-step integrator, the number of steps taken by the propagation must remain the same over the range of α 's (see [Section 5.2.2.3](#)).

5.1.3.2 Second-Order Finite Differences

[Equation \(5.30\)](#) can be slightly modified to obtain the coefficients for second-order derivatives:

$$\begin{bmatrix} 1 & \dots & 1 \\ \alpha_1 & \dots & \alpha_{n+2} \\ \alpha_1^2 & \dots & \alpha_{n+2}^2 \\ \dots & \dots & \dots \\ \alpha_1^{n+1} & \dots & \alpha_{n+2}^{n+1} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \\ \beta_{n+2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ \dots \\ 0 \end{bmatrix} \quad (5.31)$$

For multivariate functions, the gradient is obtained by solving the system of [Eq. \(5.30\)](#) and applying [Eq. \(5.25\)](#) on each dimension of the state. Each element is obtained by applying one derivative approximation after the other ($\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial \frac{\partial f}{\partial x}}{\partial y}$). Therefore, the second-order derivative of a function of two variables $f(x, y)$ can be approximated by:

$$\frac{\partial f(x, y)}{\partial x \partial y} = \frac{1}{h^2} \sum_{i=1}^{n+1} \sum_{j=1}^{n+1} \beta_i \beta_j f(x + \alpha_i h, y + \alpha_j h) + \mathcal{O}(h^{n-1}) \quad (5.32)$$

In the following, the derivatives computed using [Eq. \(5.25\)](#) and [\(5.32\)](#) will be called generic finite differences, and are computed with $n = 4$. The

corresponding stencil and coefficients are:

$$\alpha = [-2 \quad -1 \quad 0 \quad 1 \quad 2] \quad (5.33)$$

$$\beta = \left[\frac{1}{12} \quad -\frac{2}{3} \quad 0 \quad \frac{2}{3} \quad -\frac{1}{12} \right] \quad (5.34)$$

When using a variable-step integrator, the integration path and the step sizes will change for each propagation of a perturbed X . Therefore, the finite differences methods capture the changes due to the step size function (see [Eq. \(5.9\)](#) and [Section 5.2.2.2](#)). Similar to the CSD approximation, it is trivial to take advantage of the symmetry of the second-order STM [[Eq. \(2.16\)](#)] when using finite differences, and the implementation used in this work computes only the necessary unique terms.

The magnitude of the perturbation magnitude h is critical for the accuracy of finite difference methods. In order to obtain the most accurate partials for this chapter, the perturbation magnitude is tuned using a process described in [Section 5.2.1.4](#).

5.2 Applications

In order to demonstrate the subtle differences between the three methods presented in [Section 5.1](#), several applications are implemented and the results are analyzed. The test cases and applications are explained, and different types of numerical results are presented. First, the differences between fixed- and variable-step integration are highlighted. Then, the accuracy and computational efficiency of all methods are compared.

All integrations are performed using explicit integration routines: the variable-step integrator used is a Runge-Kutta-Fehlberg integrator of order 8 with embedded order 7 (RKF(7)8), and the fixed-step integrator is a Runge-Kutta-Fehlberg integrator of order 8 (RKF8). Note however, that other integration schemes (such as some implicit or predictor-corrector methods) also utilize multiple segments and variable-step sizes to adaptively capture the dynamics. Therefore, the general conclusions of the tests below are expected to apply more broadly to other variable-step integrators.

All simulations in this study use the Intel Visual Fortran compiler XE 12.1.6 on a Windows 7 workstation running on a single core of a quad-core Intel Xeon W3550 CPU with 3.07GHz clock speed, and 24GB of RAM.

5.2.1 The Testing Framework

5.2.1.1 Test Cases

The set of test cases consists of three unstable three-body periodic orbits, and two two-body orbits. For the three-body orbits, the initial conditions for the primary and secondary (in units of LU and TU) are:

$$[x_0 \quad y_0 \quad z_0 \quad \dot{x}_0 \quad \dot{y}_0 \quad \dot{z}_0]_{m_1}^T = [-\mu \quad 0 \quad 0 \quad 0 \quad -\mu \quad 0]^T \quad (5.35)$$

$$[x_0 \quad y_0 \quad z_0 \quad \dot{x}_0 \quad \dot{y}_0 \quad \dot{z}_0]_{m_2}^T = [1 - \mu \quad 0 \quad 0 \quad 0 \quad 1 - \mu \quad 0]^T \quad (5.36)$$

where μ is the dimensionless mass ratio $\frac{m_2}{m_1+m_2}$, and the typical equations of motion are described by Szebehely [319]. The equations of motion are normalized for numerical conditioning purposes. The initial conditions of the three-body periodic orbits are given in [Table 5.1](#), and the trajectories are plot-

Table 5.1: Initial conditions for the 3-body periodic orbits

	PO 1	PO 2	PO 3
x_0 (LU)	0.8982275190	1.0486505808	0.974785880
y_0 (LU)	0.0	0.0	0.0
z_0 (LU)	0.0	0.0	0.0712951519
\dot{x}_0 (LU/TU)	0.0	0.0	0.0
\dot{y}_0 (LU/TU)	-0.01806028472	-0.09354853663	-0.526306975
\dot{z}_0 (LU/TU)	0.0	0.0	0.0
T_P (TU)	25.13898226	70.53945041	2.5177274065
μ	2.528009215 E-5	2.528009215 E-5	0.01215509906
λ_{\max}	2.468621114	2.804814246 E7	17.632688124
$\ \Phi^2\ $	1.881880182 E4	8.290465853 E17	5.182997297 E6

All periodic orbits are found using differential correction

ted in Fig. 5.1. The λ_{\max} value is the largest eigenvalue of the monodromy matrix (first-order STM applied over a full period), and is used as a proxy for the degree of sensitivity of the trajectory. High values of λ_{\max} indicate highly sensitive orbits ($\lambda_{\max} > 1$ is a formal indication of instability [110, 50]). The norm of the second-order STM Φ^2 is further given as an indication of nonlinearity. The sensitivity and nonlinearity of the dynamics are important in this section as they impact the quality of the derivatives. Gradient-based solvers working with such highly sensitive trajectories are limited to taking very small steps from iteration to iteration, thereby accentuating the importance of smoothness and accuracy of the model sensitivities. As can be noted on Table 5.1, PO2 is several orders of magnitude more sensitive and nonlinear than the other three-body orbits, due to multiple close fly-bys of the secondary.

The two-body orbits both have the same initial conditions (with units

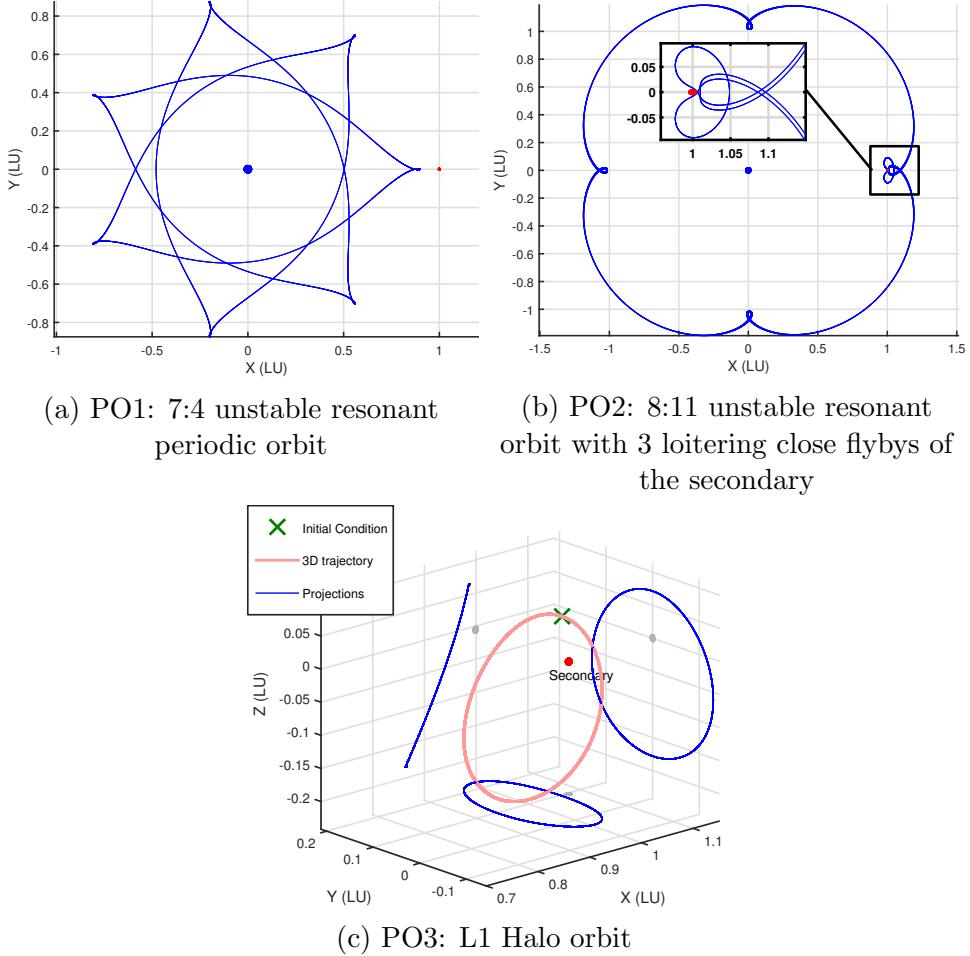


Figure 5.1: Three-body periodic orbits test cases

of LU and TU):

$$[x_0 \quad y_0 \quad z_0 \quad \dot{x}_0 \quad \dot{y}_0 \quad \dot{z}_0]^T = [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]^T \quad (5.37)$$

The standard gravitational parameter is $Gm = 1 \text{ LU}^3/\text{TU}^2$, and the typical equations of motion are found in [Section 2.2.1](#). TB1 is a single revolution of a circular Keplerian orbit (period $t_f = 2\pi \text{ TU}$). TB2 has a perturbing thrust

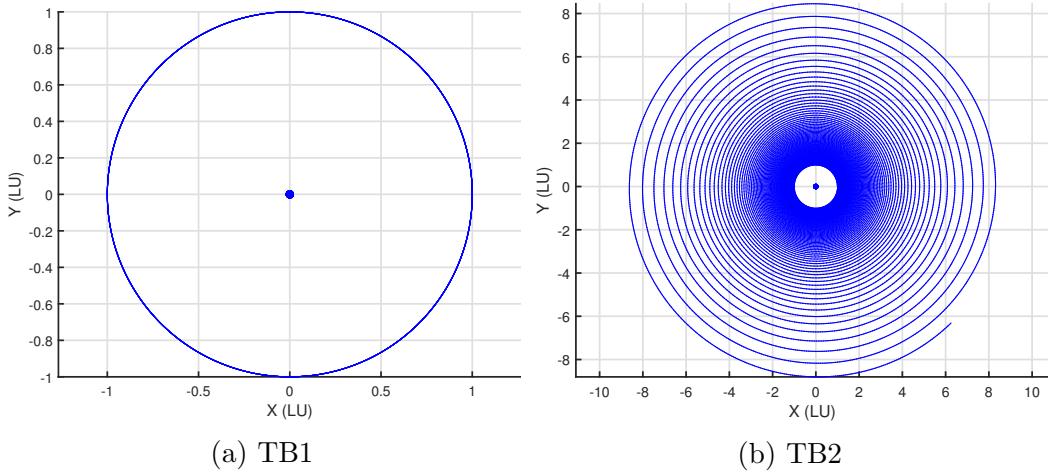


Figure 5.2: The two-body orbits

along the velocity vector:

$$\mathbf{a}_p = a \frac{\mathbf{v}}{rv} \mathbf{L} \mathbf{U} / T \mathbf{U}^2 \quad (5.38)$$

where \mathbf{v} is the velocity vector, v is the velocity, r is the distance to the attracting body, and a is the acceleration and equals to $7.10^{-4} \text{LU}^2/\text{TU}^2$. TB2 is propagated for $t_f = 900\pi$ TU, corresponding to approximately 100 revolutions of the spiraling orbit. TB2 is several orders of magnitude more sensitive than TB1. The largest eigenvalue of the first-order STM for TB2 is $\lambda_{\max} = 8.9810.10^3$, whereas it is $\lambda_{\max} = 1$ for TB1. The norms of the second-order STMs are $\|\Phi^2\| = 1.0396.10^3$ and $\|\Phi^2\| = 3.3547.10^7$ for TB1 and TB2 respectively.

5.2.1.2 Fixed Path Feature

A number of observations and recommendations in this chapter concern the use of variable-step integration. In order to highlight some of the

differences between fixed- and variable-step integration, a “fixed path” feature is implemented in the integration routines: if turned on, it records the step sizes taken by a reference variable-step propagation. Those step sizes can then be used for subsequent integrations, using the fixed-step RKF8. This feature also allows to simulate fixed-step integration while following a path decided by a variable-step integrator, and therefore achieving better precision, since the step sizes are adapted to the sensitivity of the reference trajectory at each point. It should be noted that this solution is used in the present chapter in order to comparatively study fixed- and variable-step propagation of sensitive orbits (with close flybys), while removing the need for time regularizations. In practice, fixed-step integration should be used in conjunction with regularized equations of motion.

5.2.1.3 The Linesearch Application

The linesearch application is used in order to demonstrate the effects of the different integration and derivation methods in an optimization context. In gradient-based optimization methods, the Jacobian and Hessian matrices are used to estimate the direction of descent and curvature of the function to be optimized [127, 109]. If the partials do not accurately predict the change in the state due to a perturbation, the optimization algorithm will follow sub-optimal descent directions, and may struggle converging.

In this context, the accuracy of the STMs is measured with respect to function evaluations in the vicinity of the initial point. The trajectory and its STMs are evaluated from an initial point \mathbf{X}_0 to a final point $\mathbf{X}_{f,0}$. Then,

a series of small $\delta\mathbf{X}$ are applied to the initial state, and the trajectory is propagated again - usually in a state only mode to save computation time - to obtain the resulting final state. These final states along the linesearch are also approximated using the STMs from the initial reference trajectory, to first- and second-order:

$$\mathbf{X}_{f, \text{expected}}^1 = \mathbf{X}_{f,0} + \Phi^1(t_0, t_f)\delta\mathbf{X} + \mathcal{O}(\delta X^2) \quad (5.39)$$

$$\mathbf{X}_{f, \text{expected}}^2 = \mathbf{X}_{f,0} + \Phi^1(t_0, t_f)\delta\mathbf{X} + \frac{1}{2}\delta\mathbf{X}^T \bullet_2 \Phi^2(t_0, t_f)\delta\mathbf{X} + \mathcal{O}(\delta X^3) \quad (5.40)$$

The error at each point in the linesearch is computed as the relative norm of the difference in the final states:

$$\text{Err} = \frac{\|\mathbf{X}_{f, \text{expected}} - \mathbf{X}_{f, \text{real}}\|}{\|\mathbf{X}_{f, \text{real}}\|} \quad (5.41)$$

The results of the application are plots of this error for a range of $\delta\mathbf{X}$'s, and for different scenarios (integrator, partials computation method). The error is usually computed with respect to the second-order expected state of Eq. (5.40), except where indicated. In this section, $\delta\mathbf{X}$ is along the vector $[1 \ 1 \ 1 \ 1 \ 1 \ 1]^T$, and the range of the perturbations is $\delta X \in [10^{-12}, 10^{-2}]$, except where otherwise indicated. The plots therefore show the accuracy of the approximation as well as the influence of the truncation error. Note that for small δX , the accuracy is almost always on the order of 10^{-15} , because when δX is small, only the first few digits of the STMs matter. Such small δX would not be used in practice as it would yield a very large number of iterations for a root-solving or optimization algorithm. In a similar manner, when δX becomes large, the truncation error becomes the dominant term in the error, and all

methods yield approximately the same error. When using the Root Mean Square (RMS) of the linesearch results (for conciseness purposes), the range is chosen to be $\delta X \in [10^{-12}, 10^{-7}]$, in order to not include truncation error.

This application can be used to estimate the error encountered in a linesearch algorithm (hence its name), or from one iteration to another in an optimization algorithm. A key component of a linesearch in practice is that the propagations are performed in a state only mode, i.e., derivatives are not computed at every point along the search. However, a second case is relevant in algorithms that don't perform traditional linesearches, but compute partials at every iteration of a search. In this second case, the simulated linesearches represent major iterations of an optimization algorithm.

5.2.1.4 Tuning the Perturbation Magnitude

The perturbation magnitude h plays an important role in the accuracy of finite difference methods, because of the combined effects of truncation error and of floating point operations [156, 115, 152]. Tuning methods have been developed to find the ideal perturbation magnitude for a specific finite difference formula and a specific problem [279]. The tuning process represents additional complexity and compute time, and has to be repeated when changing the trajectory. Therefore, heuristic methods can be employed when the practitioner does not desire to re-tune the algorithm for every different problem. The main goal is to balance the effects of truncation error and of the floating point arithmetic errors, since truncation error decreases when h decreases, whereas a small h leads to higher error in floating point opera-

tions. A simple heuristic, valid under strong assumptions, was developed in Ref. [247]. However, the assumptions do not hold for most space trajectories, and a tuning process is used in the present chapter. The linesearch application is used: the RMS of the linesearch error for $\delta X \in [10^{-12}, 10^{-7}]$ is computed for a range of perturbation magnitudes, and the value resulting in the smallest error is chosen. In order to decouple the tuning of the perturbation from the problems highlighted in Section 5.2.2, the tuning process uses fixed-path for both the reference and the linesearch propagations. The results of the tuning process are plotted in Fig. 5.3 for two of the considered orbits. Note that the x axis of Fig. 5.3a and 5.3b is the magnitude of the perturbation applied to obtain the first- and second-order STMs (and not the perturbation applied in the linesearch application). The results of Fig. 5.3 and Table 5.2 demonstrate that larger perturbation magnitudes should be applied when using a higher-order finite difference method. Moreover, as expected, the tuned perturbation magnitude is smaller when the orbit is more sensitive. The integration tolerance did not seem to influence the ideal perturbation magnitude. For the rest of this study, the values given in Table 5.2 are used for h .

5.2.2 The Caveats of Variable-Step Integration

When using variable-step integrators to propagate a trajectory, the integration path is decided on the fly by the integrator. The step sizes depend on the error estimates at each step, which vary depending on the equations of motion, on the initial conditions, and on the roundoff error from the dynamics evaluations. This section highlights the consequences of a changing integration

Table 5.2: Perturbation magnitude for the finite difference methods

	Forward	Central	Generic
PO1	10^{-8}	10^{-7}	10^{-6}
PO2	10^{-9}	10^{-8}	10^{-8}
PO3	10^{-9}	10^{-8}	10^{-7}
TB1	10^{-8}	10^{-7}	10^{-6}
TB2	10^{-8}	10^{-6}	10^{-5}

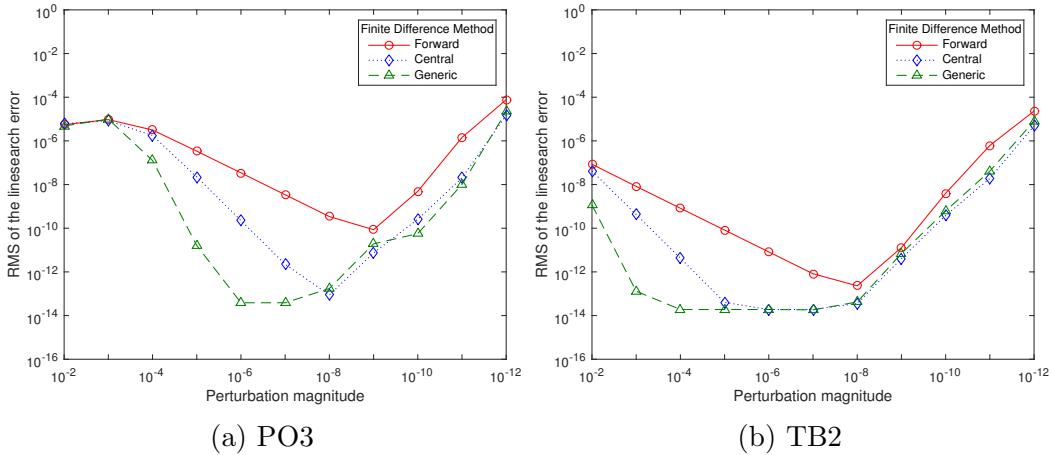


Figure 5.3: Accuracy results for $\epsilon = 10^{-3}$, when varying the perturbation magnitude used in the finite difference methods

path on the accuracy of the partials computation.

5.2.2.1 Path with Variational Equations

When propagating the variational equations alongside the EOMs, the variable-step integrator bases its error estimates on a system of dimension $N + N^2 + N^3$. In the following, the $N + N^2 + N^3$ vector containing the state and STMs is called the augmented state. Propagating the augmented state

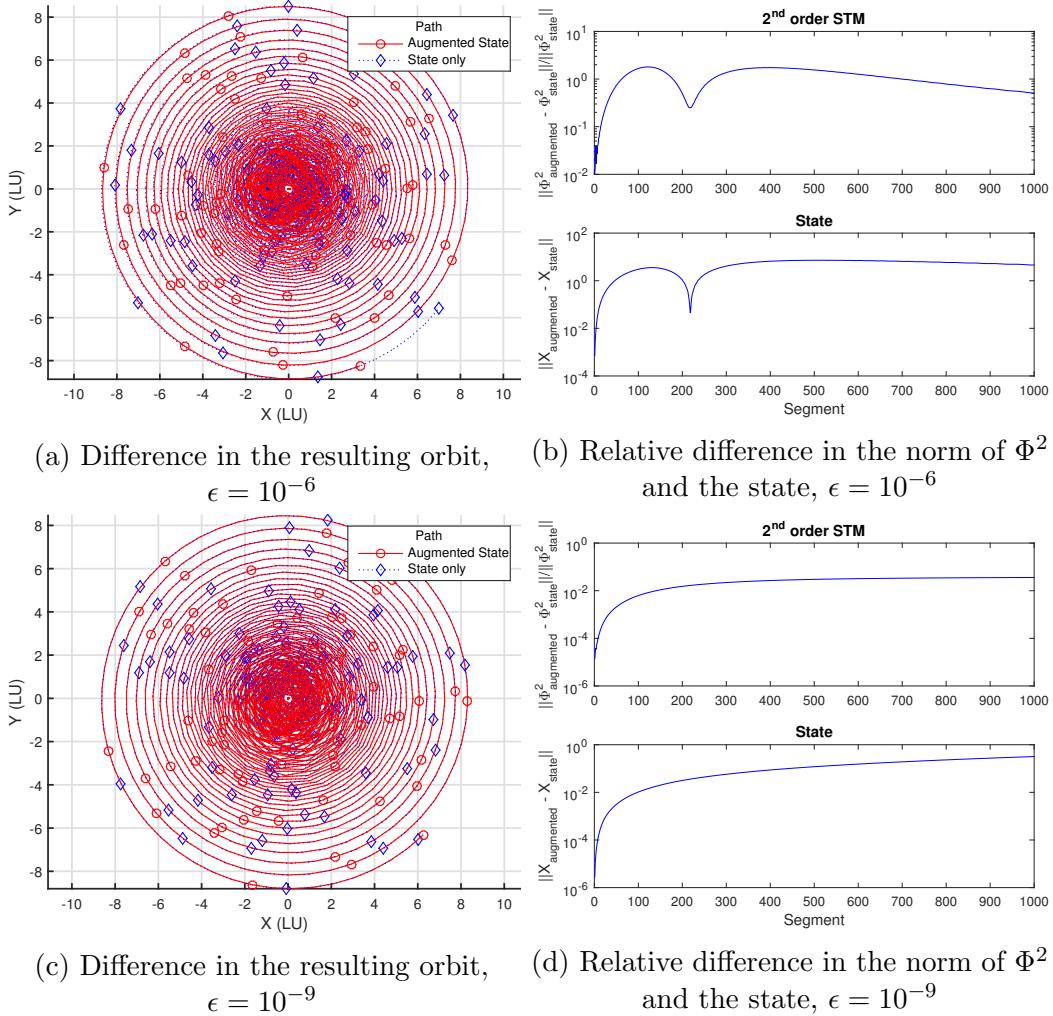


Figure 5.4: TB2 and its sensitivities integrating the state only path or the state and variational equations path

generally yields a greater number of integration steps than when integrating the state alone (dimension N), creating two different integration paths. The difference in the resulting trajectories for TB2 is shown on Fig. 5.4a and 5.4c. The effect of the different paths on the state and STMs can be observed on Fig. 5.4b and 5.4d: the fixed path feature is used to integrate the STMs using

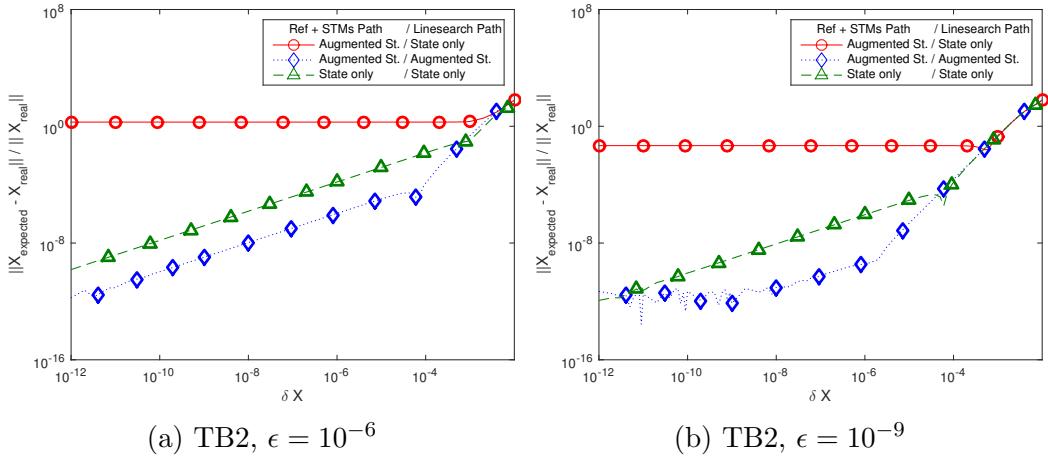


Figure 5.5: Results of the linesearch application for TB2, using the RKF(7)8 and the variational equations

the state only path and the augmented state path, which generates two different estimations of the state and sensitivities. For the variational equations, using the fixed path feature is equivalent to restricting the estimation of the error to the state variables only, as the integrator now follows the state only path. Figure 5.5 shows the effects of the mismatch on the linesearch results for the same orbit. Both figures, as well as the other linesearch application plots in the rest of the chapter show the transition to truncation error: the converging lines on the right-hand side of the plots correspond to truncation error becoming dominant.

Figure 5.4 and Fig. 5.5 demonstrate that state only propagations and augmented state propagations should not be mixed. Many nonlinear optimization methods require both gradient information (augmented state propagations) to determine a search direction, and function evaluations (state only propagations) for the linesearch. In such cases, a common integration path

should be chosen, otherwise the partials are inconsistent. This common path can be efficiently accomplished by: 1) fixing the path (using a fixed-step integrator or a fixed path feature) or 2) evaluating the integration error on the state only when propagating the augmented state (triangle case in Fig. 5.5).

Figure 5.4 and Fig. 5.5 also demonstrate that the difference in path is more important for low accuracy propagations. For higher accuracy propagations, the trajectory approximated by the integration converges towards the physical solution of the equations of motion. Therefore, the potential integration paths present less dispersion, and the smaller differences in said paths have less influence on the approximation. Choosing an appropriately tight integration tolerance is a way to mitigate the problems due to different integration paths. Note that the differences in path are exacerbated for highly sensitive orbits, such as TB2 used in Fig. 5.4 and Fig. 5.5. Propagating the augmented state at all times produces more accurate results than using the state only path since it essentially corresponds to tightening the tolerance of the variable-step integrator.

5.2.2.2 Partials of the Variable-Step Integration

The initial conditions of the integration influence the integration path, which in turn changes the final computed state. This dependency is explicitly expressed in Eq. (5.9). The $\delta t(\mathbf{X})$ function implemented in the integrator used for the present study is shown in Eq. (5.11), and the corresponding partial with respect to the state is Eq. (5.12).

As is observed formally in Section 5.1.1, the variational equations do

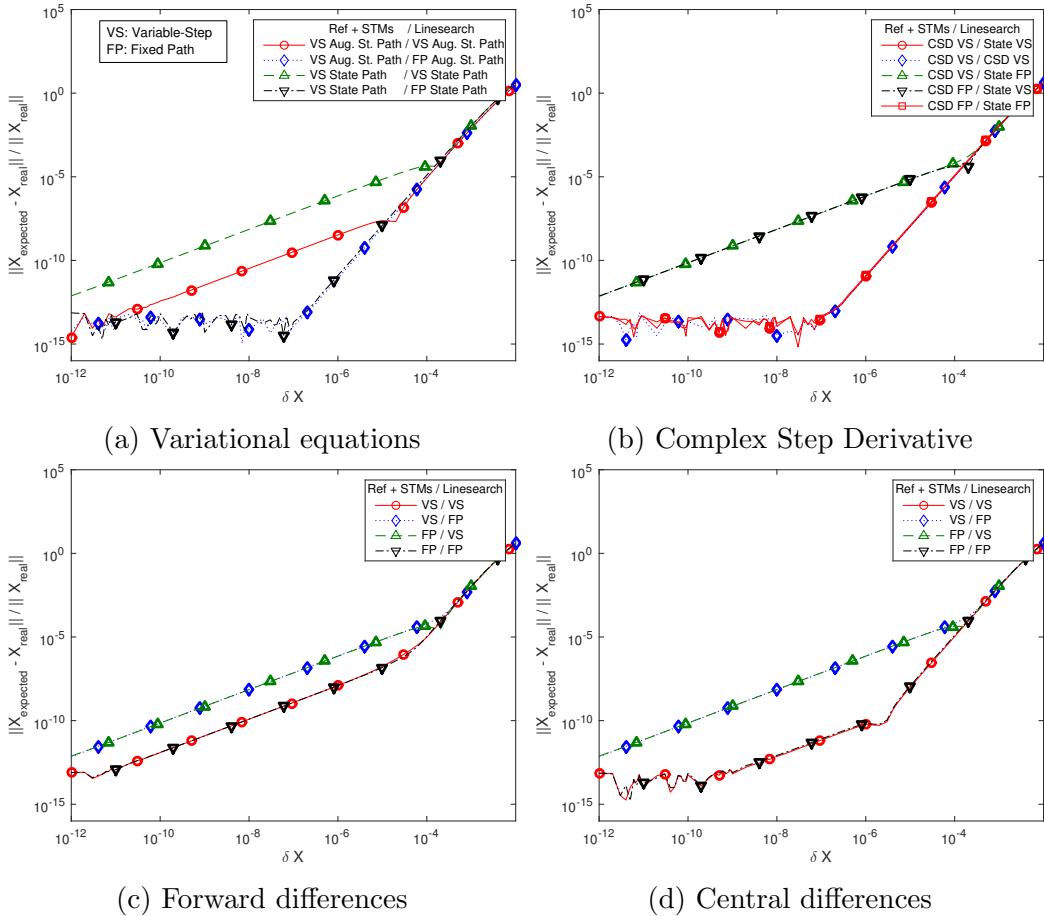


Figure 5.6: Linesearch results for PO3 with $\epsilon = 10^{-3}$

not capture the extra term in the partials due to $\frac{\partial \delta t}{\partial \mathbf{x}_0}$. The CSD and finite difference methods are expected to take this change into account. This section confirms these findings.

Figure 5.6 presents the results of the linesearch application for PO3, and an integration tolerance $\epsilon = 10^{-3}$. A number of different configurations are analyzed, mixing the use of variable-step and fixed path integrations (noted VS and FP in the legends, respectively). Figure 5.6a demonstrates that when

using the variational equations, the STMs predict the change in the final state much more accurately when the state propagations use a fixed path. When the variable-step is used for the linesearch, part of the change is not accounted for, as is predicted in [Section 5.1.1](#). It is also worth noting that once more, the error is smaller when the path is computed from the augmented state, which is explained by the δt_i term in [Eq. \(5.12\)](#): the smaller the step sizes, the smaller the $\frac{\partial \delta t}{\partial \mathbf{X}_0}$ term.

The CSD is analyzed in [Fig. 5.6b](#). Poor approximations of \mathbf{X}_f happen only when fixed and variable paths are mixed. Since the CSD algorithm generates partials of the whole integration process, it captures the $\frac{\partial \delta t}{\partial \mathbf{X}_0}$ term when using variable-step integration, yielding STMs that can predict the changes due to the variation of the path.

Finally, the ability of the finite difference methods to capture the $\frac{\partial \delta t}{\partial \mathbf{X}_0}$ term is demonstrated in [Fig. 5.6c](#) and [Fig. 5.6d](#). Once again, mixing fixed and variable paths yields worse results than those with consistent paths. Note that using the fixed path feature for the finite difference methods means that every propagation of a perturbed state within the partials computation will follow the reference path. The better accuracy of the central differences over the forward differences also is exhibited.

For all methods, the errors observed when mixing variable-step and fixed-path integrations are identical, corresponding to the error introduced by the extra $\frac{\partial \delta t}{\partial \mathbf{X}_0}$ term, which is shown to be the dominant error in those cases. Note that the apparent chatter encountered for lower δX and errors around

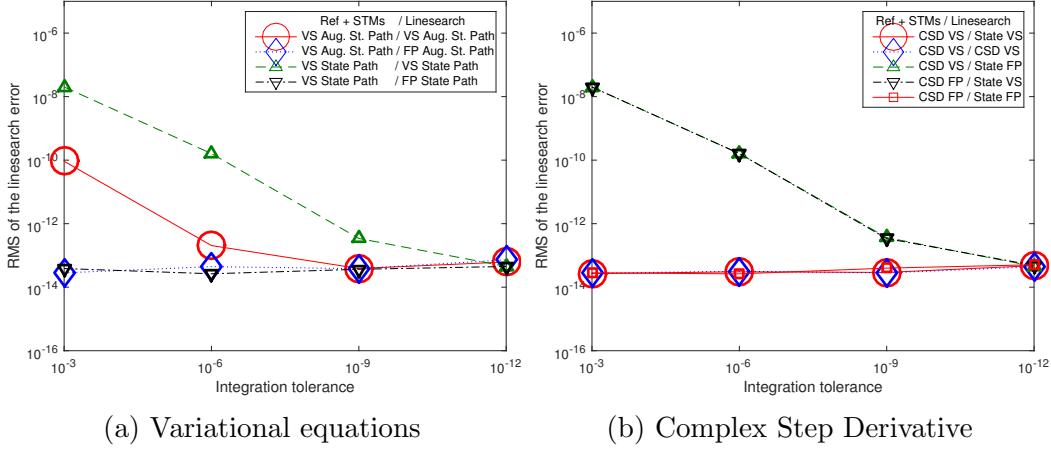


Figure 5.7: RMS of the linesearch results for PO3

10^{-15} is due to floating point operations error in the computation of the error, since $\mathbf{X}_{f,\text{expected}}$ and $\mathbf{X}_{f,\text{real}}$ are extremely close.

All the cases presented in Fig. 5.6 are run for different tolerances. For conciseness purposes, the RMS of the error for $\delta X \in [10^{-12}, 10^{-7}]$ is chosen to represent the accuracy of each configuration. The results are presented in Fig. 5.7, which demonstrates that the influence of the $\frac{\partial \delta t}{\partial \mathbf{X}_0}$ term decreases as the tolerance is tightened.

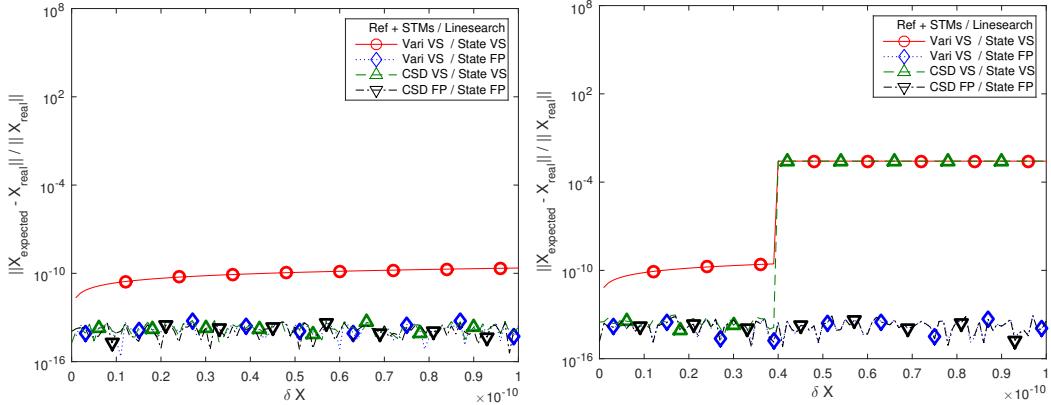
5.2.2.3 Discontinuity of the Integration Path

The variations in step size discussed in the previous sections deteriorate the quality of the partials. As demonstrated in Section 5.2.2.2, the small inaccuracies in the STMs when using variational equations and a variable step integrator can be accounted for by using the CSD and the finite difference methods. However, some of the multiple propagations of neighboring trajectories performed by the linesearch might require a different number of

integration steps than that used for the reference trajectory. This change in number of steps causes a discontinuity in the integration function with respect to the initial conditions, which affects the accuracy of all methods of partials computation. These cases happen naturally on certain trajectories, for certain integration tolerances. In this section, the initial conditions of PO3 are slightly modified to create a new perturbed trajectory where the discontinuity is apparent, and decoupled from other subtleties:

$$\tilde{X}_{0, \text{PO3}} = X_{0, \text{PO3}} + [2.053002.10^{-4} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T \quad (5.42)$$

Using the RKF(7)8 with $\epsilon = 10^{-3}$, $\tilde{X}_{0, \text{PO3}}$ is propagated from $t = 0$ to $t = T_{P, \text{PO3}}$ in 6 steps. The linesearch direction in this example is along the $[1 \ 0 \ 0 \ 0 \ 0 \ 0]^T$ vector, and the range is $\delta X \in [0, 10^{-10}]$. When the linesearch reaches $\delta X = 4.10^{-11}$, an additional step is necessary to complete the integration. [Figure 5.8](#) and [5.9](#) show the resulting linesearch results. A step function increase in the error is expected when $\delta X \geq 4.10^{-11}$, since the reference and STMs propagation can not predict the change in number of steps. In the case of the finite difference methods ([Fig. 5.9](#)), when using variable-step integration for the reference and STMs, an error over the whole domain is expected, in addition to the step function increase at $\delta X = 4.10^{-11}$. The finite differences methods use the propagation of perturbed states to approximate the partials. For this particular orbit, when computing the partials with respect to X_1 , the finite difference methods perturb the initial state along X_1 , with a magnitude greater than 4.10^{-11} (see [Table 5.2](#)), which causes the discontinuity to occur, corrupting the computation of the STMs. The fixed path finite differences do



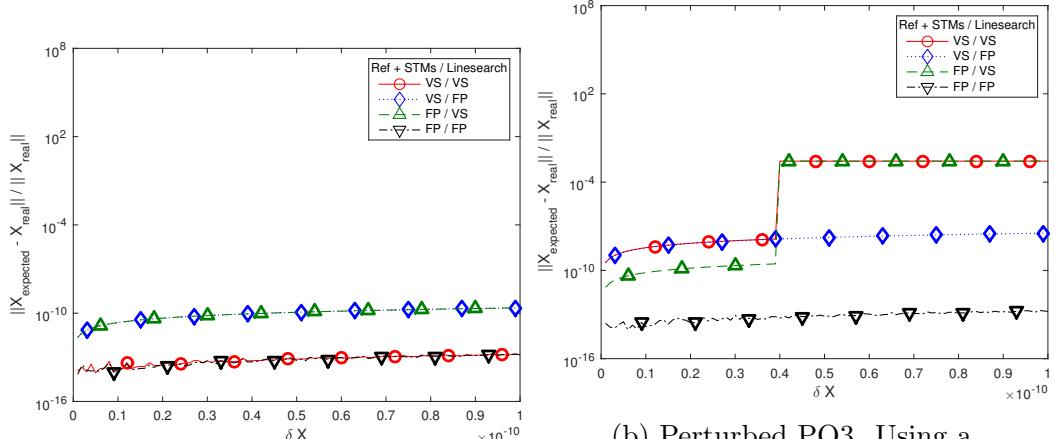
(a) Original PO3. Only the variational equations used with a variable-step linesearch exhibit the higher error explained in [Section 5.2.2.2](#).

(b) Perturbed PO3. When using a variable-step integrator for the linesearch, the change in number of steps leads to a step function increase in the error for both methods.

Figure 5.8: Results of the linesearch using the variational equations and the CSD method for PO3 and its perturbed version, with $\epsilon = 10^{-3}$

not encounter the discontinuity. The propagation of the perturbed state for the CSD approximation has very little chances to encounter a discontinuity, as the magnitude of the perturbation is extremely small ($h = 10^{-40}$).

The expected step function in the error can be observed on [Fig. 5.8b](#) and [5.9b](#), when $\delta X \geq 4.10^{-11}$. It only occurs when the linesearch propagations use the variable-step integrator, and not when the paths are fixed. All methods for partials computation are affected by this discontinuity. As expected, the discontinuity has a particularly negative effect on the accuracy of the finite differences methods. Comparing [Fig. 5.9a](#) and [Fig. 5.9b](#) shows that, in addition to the jump in error at $\delta X = 4.10^{-11}$ when the linesearch state uses the RKF(7)8, the general accuracy of the STMs over the whole linesearch is also degraded when using the variable steps. The diamond and



(a) Original PO3. Mixing variable-step and fixed path integrations for the reference and linesearch creates a higher error, explained in Section 5.2.2.2.

(b) Perturbed PO3. Using a variable-step integrator for the linesearch increases the error when encountering the change in number of steps. When using the variable-step integration for the reference and STMs (circle and diamond cases), the error is larger on the whole domain, as expected

Figure 5.9: Results of the linesearch using the central difference method for PO3 and its perturbed version, with $\epsilon = 10^{-3}$

circle cases of Fig. 5.9b have a worse accuracy than in Fig. 5.9a over the whole domain, because the computation of the STMs is corrupted by the discontinuity. Once again, these effects are exacerbated for looser integration tolerances. As mentioned in Section 5.2.2.1, tighter tolerances yield less dispersion of the integration path, and therefore a smaller discontinuity of the resulting integration.

5.2.3 Accuracy and Timings of the Partial Computations

In this section, the methods of Section 5.1 are compared in terms of accuracy and computational time. The accuracy is measured as the RMS of

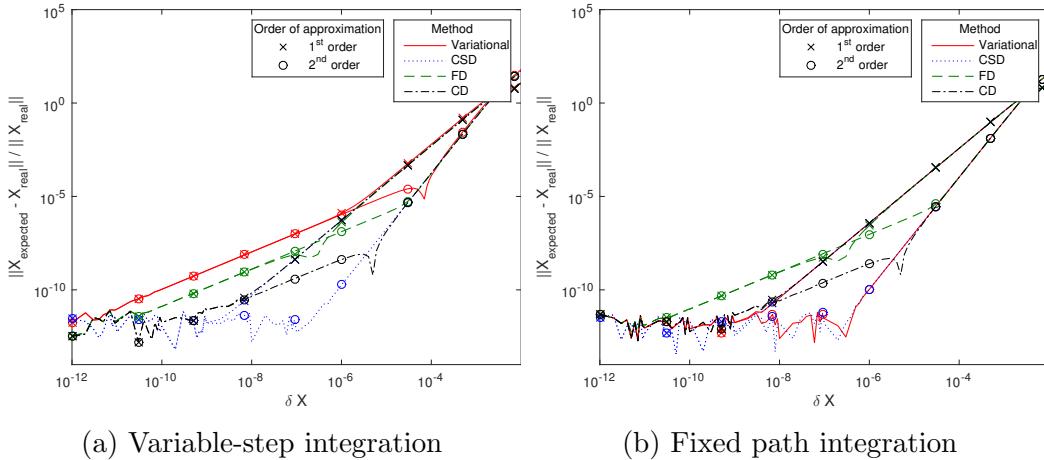


Figure 5.10: Linesearch results for fixed and variable-step integration of TB2, $\epsilon = 10^{-6}$

the errors in the linesearch application over the range $\delta X \in [10^{-12}, 10^{-7}]$. The methods are timed using Fortran’s intrinsic routine `SYSTEM_CLOCK`, and the timings are averaged over 100 runs of the STMs computations. Parallel implementations of the STMs are not considered.

As is usually the case in optimization algorithms, fixed- and variable-step integration are not mixed in the linesearch application, and both the reference trajectory/STMs and the linesearch states are propagated using the same type of integrator. The fixed path feature is used to simulate fixed-step integration. In light of the pitfalls of variable-step integration discussed in Section 5.2.2, only some of the cases shown above are selected. When propagating the variational equations, the variable-step integration follows the augmented state path, and the fixed path integration follows the state only path.

Figure 5.10 shows the fixed- and variable-step results for TB2 and

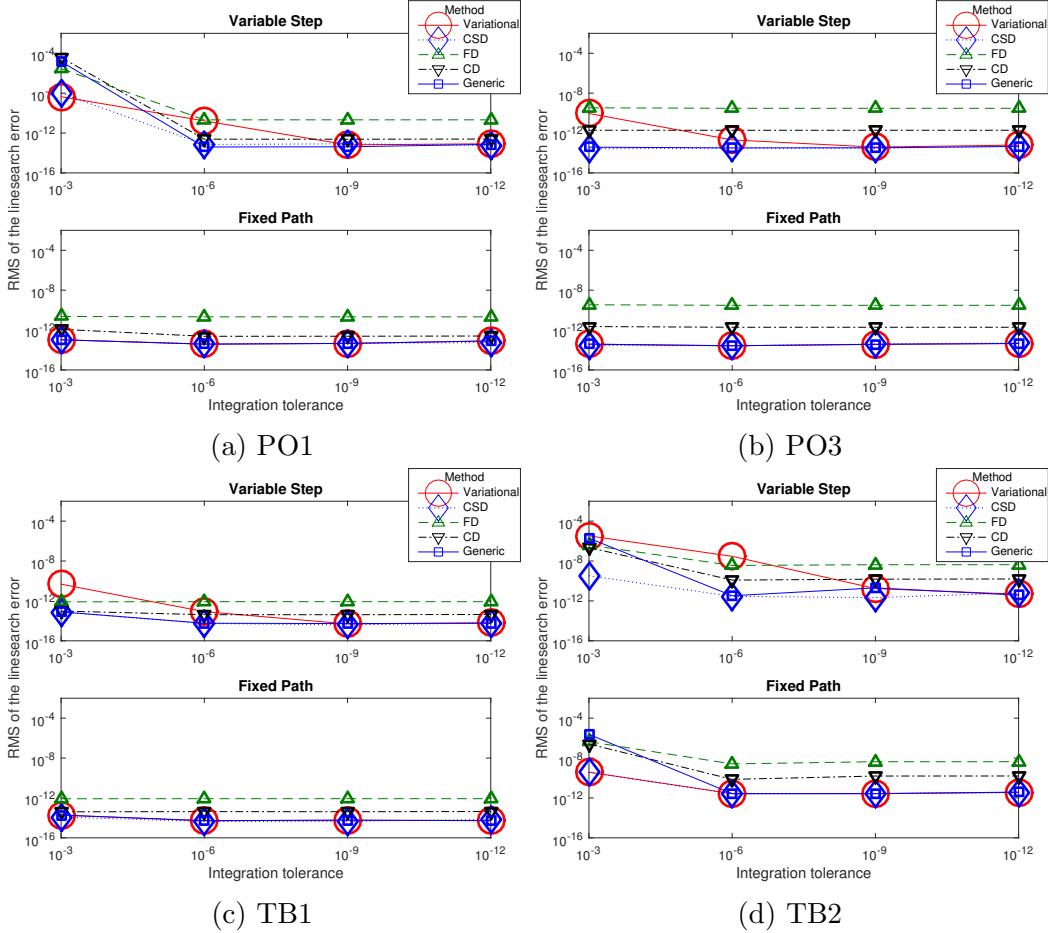


Figure 5.11: Accuracy

$\epsilon = 10^{-6}$ along the linesearch range. The methods are compared over the whole range of δX . The variational equations are outperformed by all other methods when using variable-step integration, because of the mixing of integration paths. Note that $\epsilon = 10^{-6}$ is a typical integration tolerance (it is the default for Matlab integrators for instance), highlighting once more that the mixing of integration paths presented in Section 5.2.2.1 is a particularly bad practice. Figure 5.10 also shows the results obtained with the first-order only

approximation of the state, and highlights the decrease in truncation error when using second-order information.

[Figure 5.11](#) presents the accuracy results for the different trajectories for all considered tolerances. Both [Fig. 5.10](#) and [Fig. 5.11](#) confirm the results presented in [Section 5.2.2](#). When using variable-step integration, the CSD method achieves a better accuracy than the variational equations, especially for loose integration tolerances. This confirms the influence of the $\frac{\partial \delta t}{\partial \mathbf{x}}$ term on all orbits (see [Section 5.2.2](#)). The CSD approximations are also only slightly slower to compute than the variational equations when a variable-step integrator is used (see [Fig. 5.12](#)), since the propagation of the augmented state usually takes a larger number of integration steps. When using a fixed path, both methods have equivalent accuracies. Moreover, the variational equations are faster to propagate than the CSD derivatives, since their propagation now follows the state only path, and takes the same number of steps as the CSD.

The common finite difference methods present a loss in accuracy. The forward difference is the least accurate method for all cases, but is also the fastest, highlighting the common trade-off between accuracy and computational speed. The central differences improve the predicted final state, and even outperform the variational equations in certain loose

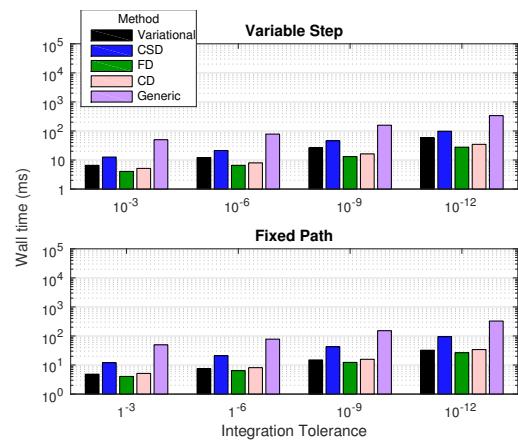


Figure 5.12: Timings of the different methods for TB2

tolerance cases. The central differences do not attain the same accuracy as the CSD. In most of the examples presented here, the generic finite differences (i.e., using five evaluations for every partial) achieve similar accuracy to the variational equations and CSD approach, but have a high compute time penalty (and their step sizes have to be custom tuned to obtain such accuracy). All the results in this section are exacerbated for low accuracy propagations. For tighter integration tolerances, only the most sensitive orbits are affected by the findings of [Section 5.2.2](#).

Finally, it is important to note that the parallel computation of the sensitivities is possible. The timings could therefore be improved if implemented in parallel (for instance using OpenMP). The parallel implementation of CSD or finite differences is straight-forward, since each propagation of a perturbed state is independent from the others. The variational equations can be computed in parallel as well, but the implementation is not trivial.

The Special Case of PO2 The results obtained for PO2 are presented separately from the other test cases, in [Fig. 5.13](#). PO2 is the most sensitive of the orbits tested in this chapter, and therefore, the range for the RMS is $\delta X \in [10^{-12}, 10^{-9}]$, as truncation error becomes dominant for larger values of δX . The error in the linesearch application invariably grows as the integration tolerance is tightened, because the global nature of the orbit changes with the integration tolerance. For low accuracy propagations, the integration error is such that the phasing of the spacecraft and of the secondary body is corrupted, and the number of close flybys of the secondary is less than for tight integration

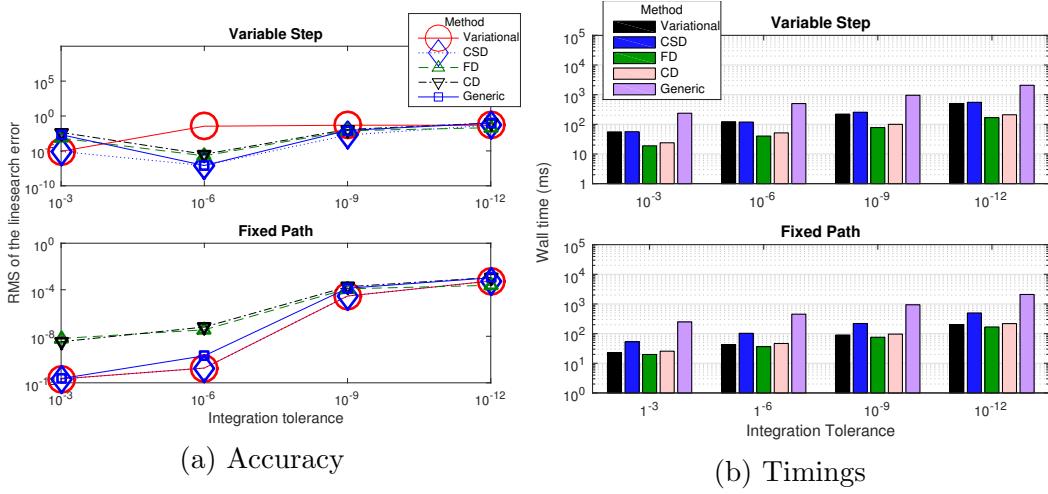


Figure 5.13: Accuracy and timings results for PO2

tolerances. The sensitivity of the trajectory and therefore the magnitude of the STMs increases with the number of flybys, resulting in an increase of the associated error. However, for a given integration tolerance, the results discussed in Section 5.2.2 remain valid. For the variable-step integration and certain integrations tolerances, the CSD algorithm outperforms the variational equations by several orders of magnitude, confirming that the sensitivity of the orbit exacerbates the importance of the $\frac{\partial \delta t}{\partial \mathbf{x}}$. PO2 is a good example of a case where using a loose integration tolerance for preliminary design would not be suited, since tighter tolerances result in very different trajectories.

5.3 Conclusions

In this chapter, three common methods of partials computation are examined in detail. The propagation of the variational equations is compared to the complex-step derivative approximation, and finite difference methods.

The accuracy of the first- and second-order partials computed by each method is analyzed, for fixed- and variable-step integrators, and subtleties of the partials computation are documented. The integration path is critical: when using sensitivity analysis for a linesearch, the integration path for the partial computation and the linesearch has to be consistent.

In this light, three main pitfalls of variable-step integration for STMs computation are highlighted. 1) When propagating the variational equations, it is important to avoid mixing propagations of the state only and of the state and STMs. 2) Variable-step integration can lead to discontinuities of the integration function, due to the step size selection changing the number of steps required to complete the integration. These discontinuities can not be predicted by the partials computations. Moreover, the finite difference approximations are especially vulnerable to these discontinuities unless the number of integration steps is fixed when computing the perturbed trajectories. Therefore, the use of fixed-step integration and time regularization techniques is recommended to avoid the need for a variable-step integrator. 3) The variation of the step sizes has an impact on the final state, and should be accounted for in the partials. The propagation of the variational equations does not capture this change, in contrast to the cases of the CSD approximations and the finite difference methods. When using variable-step integration, the variational equations are usually outperformed by CSD, and even by the finite differences methods for loose integration tolerances. These caveats all have a larger influence for highly sensitive orbits, and for low accuracy propagations, making them especially important for the preliminary design of modern space missions

(when a low accuracy propagation is adapted and does not change the nature of the orbit).

When using fixed-step integration or variable-step integration with a tight tolerance, the accuracy of the variational equations and the CSD method are comparable. The variational equations usually are faster, especially for fixed-step integration. As expected, the forward difference method is the least accurate method of all the finite difference methods. The central differences usually improve the accuracy, without matching that of the variational equations or CSD method. The generic finite difference method can match the accuracy for nonsensitive orbits and tight integration tolerances, but has a large compute time penalty. Note that the accuracy of all finite difference methods are dependent upon choosing an appropriate perturbation step size, which is accomplished by tuning the perturbation magnitude for each method and each orbit. Although the CSD and variational equations methods usually outperform them, finite difference methods are sometimes the only available method when the dynamics require the use of black-box codes, such as linear algebra packages. The computational time necessary for the finite difference and CSD methods can be improved by a straight-forward parallel implementation of the algorithm, whereas the parallel implementation of the variational equations is much more involved.

Chapter 6

F and G series solutions to the Kepler, Stark, and Circular Restricted Three-Body Problems

6.1 Introduction

The numerical propagation of differential equations is essential to the transcription of optimal control problems into nonlinear programs. The present chapter¹ introduces a new formulation of the equations of motion (EOMs) for two dynamical systems commonly used for the approximation of spacecraft trajectories, namely the Stark and CRTB problems (see [Section 2.2](#)). Recursion equations for the F and G Taylor series method extended to these problems, are presented in [Section 6.2](#), and are implemented to high order using symbolic manipulation software. The recursion equations include a possible time regularization technique based on the Sundman transformation. [Section 2.2.3](#)

¹Work from this chapter was published in a peer-reviewed journal as:

- **Etienne Pellegrini**, Ryan P. Russell, F and G Series Solutions to the Kepler and Stark Problems with Sundman Transformations, *Celestial Mechanics and Dynamical Astronomy*, Vol. 118 No. 4, pp. 355-378, March 2014, [doi:10.1007/s10569-014-9538-7](https://doi.org/10.1007/s10569-014-9538-7)

Work from this chapter was presented as:

- **Etienne Pellegrini** and Ryan P. Russell, F and G Taylor Series Solutions to the Stark Problem with Sundmann Transformations, Paper AAS 13-725, *AAS/AIAA Astrodynamics Specialists Conference*, Hilton Head, SC, August 2013
- **Etienne Pellegrini** and Ryan P. Russell, F and G Taylor Series Solutions to the Circular Restricted Three-Body Problem, Paper AAS 14-237, *AAS/AIAA Space Flight Mechanics Meeting*, Santa Fe, NM, February 2014

presents the details of the generalized Sundman transformation and its effects on the discretization of the problem. A variety of different power laws defining the generalized Sundman transformation are considered, and their effects on the integration accuracy are presented for the F and G series, and three benchmarks integration methods. The Modern Taylor Series (MTS) presented in [Section 6.3.1.1](#) and an eighth order Runge-Kutta-Fehlberg integrator in both its fixed-step and variable-step variants (designated by RKF8 and RKF(7)8 respectively) are chosen as the benchmarks. A comparative study of the computational times required by each of the integration schemes is performed, paying close attention to compare only solutions of similar accuracy.

The numerical results presented in [Section 6.3.1](#) show that: 1) The Taylor series solutions to the Stark problem have significantly better performance than a classic RKF8 integrator; 2) The Taylor series methods for the Kepler and Stark problems are exceptionally efficient for the Sundman transformation corresponding to a unity power law, a result which departs from the conventional wisdom that the power law of 3/2 (corresponding to the so-called intermediate anomaly) is most efficient for astrodynamics; and 3) The impressive speedups of both the F and G and MTS methods compared to the RKF8 are specific to the simple form of the Stark and 2-body equations of motion.

The numerical results presented in [Section 6.3.2](#) demonstrate the successful integration of the CRTBP via F and G type series. A variable-step integrator using the F and G solution is described, and shown to have comparable performance to traditional integration methods, except for low-fidelity propagations, for which the F and G integrator with no time transformation

is demonstrated to be faster than the RKF(7)8. The F and G series solution is therefore adapted to preliminary trajectory design. The Sundman type transformations are demonstrated to improve the discretization of the orbit and lower the number of steps necessary for the propagation of the EOMs.

6.2 Development of the Series Formulations

In this section, the principles of the classic F and G Taylor series method are reviewed, and the derivation is extended to both the Stark problem and the CRTBP. The Lagrange coefficients f and g are a well-known solution method to the two-body problem [6]. The position and velocity vectors at τ are decomposed in the basis formed by position and velocity at τ_0 :

$$\begin{aligned}\mathbf{r}(\tau) &= f\mathbf{r}(\tau_0) + g\mathbf{v}(\tau_0) \\ \mathbf{v}(\tau) &= \dot{f}\mathbf{r}(\tau_0) + \dot{g}\mathbf{v}(\tau_0)\end{aligned}\tag{6.1}$$

This formulation is valid for Keplerian orbits, as the motion is confined to a plane. However, in the case of the Stark problem or the CRTBP, out-of-plane motion can occur. Therefore, at least one extra basis vector is needed to track the particle in the 3D space. The third vector necessarily has a component in the angular momentum direction, and is otherwise arbitrary. The choice for the extra basis vectors is problem dependent, and will be covered in [Section 6.2.1](#) and [Section 6.2.2](#). Moreover, in both problems, the use of the Sundman transformation introduces an extra differential equation for time, as shown in [Eq. \(2.40\)](#) and [Eq. \(2.49\)](#), and therefore requires the development of one more series to keep track of time. The methods introduced in this chapter will therefore use four or five series instead of the classic two F and G series.

However, for legacy purposes the sets of series will be called the “ F and G Stark series” and “ F and G CRTBP series”.

6.2.1 F and F Stark Series

In the Stark problem, out-of-plane motion will occur when the perturbation \mathbf{p} is out-of-plane. Therefore, a third basis vector is needed for the general 3D problem. The perturbation vector itself will be used in this study. Note that the authors considered other options for this third basis vector; the angular momentum vector and its normalized counterpart resulted in an increase in the complexity of the recursion formulas, since they are not inertially constant; any of the inertial reference frame basis vectors could not capture the full motion when the osculating orbit plane was aligned with said basis vector. The authors found that the inertially fixed perturbation vector leads to the simplest and most elegant formulation. The three vectors \mathbf{r}_0 , \mathbf{v}_0 and \mathbf{p} form a basis to the 3D space. If the perturbation is in the plane of \mathbf{r}_0 and \mathbf{v}_0 , \mathbf{p} is a redundant basis vector, but the 2D space is still spanned. This new third basis vector necessitates the development of a third series to be computed alongside F and G .

To solve the Stark integration problem using Taylor series, the development of the conventional F and G series [6] is extended. For order N , to obtain the position vector \mathbf{r} at $\tau = \tau_0 + \Delta\tau$, the following truncated Taylor series is employed:

$$\mathbf{r} \simeq \sum_{n=0}^N \frac{d^n \mathbf{r}}{d\tau^n} \Big|_{\tau_0} \frac{\Delta\tau^n}{n!} \quad (6.2)$$

As mentioned previously, \mathbf{r}_0 , \mathbf{v}_0 and \mathbf{p} form a suitable basis for either a 2D or

3D space, and therefore the \mathbf{r} derivative terms of Eq. (6.2) can be expressed as a linear combination of \mathbf{r}_0 , \mathbf{v}_0 and \mathbf{p} :

$$\frac{d^n \mathbf{r}}{d\tau^n} \Big|_{\tau_0} = F_n|_{\tau_0} \mathbf{r}_0 + G_n|_{\tau_0} \mathbf{v}_0 + H_n|_{\tau_0} \mathbf{p} \quad (6.3)$$

which, when substituted into Eq. (6.2) yields:

$$\begin{aligned} \mathbf{r} &\simeq \sum_{n=0}^N \left[F_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \mathbf{r}_0 + \sum_{n=0}^N \left[G_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \mathbf{v}_0 + \sum_{n=0}^N \left[H_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \mathbf{p} \\ &\equiv F\mathbf{r}_0 + G\mathbf{v}_0 + H\mathbf{p} \end{aligned} \quad (6.4)$$

The “extended” Lagrange coefficients form of the solution is:

$$\mathbf{r}(\tau) = f\mathbf{r}_0 + g\mathbf{v}_0 + h\mathbf{p} \quad (6.5)$$

Comparing Eq. (6.4) taken to order infinity and Eq. (6.5) yields the relation between the f , g , and h Stark functions and the F , G , and H Stark series:

$$\begin{aligned} f &= \sum_{n=0}^{\infty} \left[F_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \\ g &= \sum_{n=0}^{\infty} \left[G_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \\ h &= \sum_{n=0}^{\infty} \left[H_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \end{aligned} \quad (6.6)$$

In order to find the coefficients F_n , G_n , and H_n for any n , the successive derivatives of \mathbf{r} at τ are decomposed in the basis formed by \mathbf{r}_0 , \mathbf{v}_0 and \mathbf{p} . It is important to note that this decomposition, when evaluated at τ_0 , is the one shown in Eq. (6.3).

$$\frac{d^n \mathbf{r}}{d\tau^n} = F_n \mathbf{r} + G_n \mathbf{v} + H_n \mathbf{p} \quad (6.7)$$

Equation (6.7) is differentiated with respect to τ :

$$\frac{d^{(n+1)}\mathbf{r}}{d\tau^{(n+1)}} = F'_n \mathbf{r} + F_n \frac{d\mathbf{r}}{d\tau} + G'_n \mathbf{v} + G_n \frac{d\mathbf{v}}{d\tau} + H'_n \mathbf{p} + H_n \frac{d\mathbf{p}}{d\tau} \quad (6.8)$$

$$= F'_n \mathbf{r} + cr^\alpha F_n \mathbf{v} + G'_n \mathbf{v} + cr^\alpha G_n \left(-\frac{\mu}{r^3} \mathbf{r} + \mathbf{p} \right) + H'_n \mathbf{p} \quad (6.9)$$

Alternatively, advancing the sequence in Eq. (6.7) from n to $n+1$ leads to:

$$\frac{d^{(n+1)}\mathbf{r}}{d\tau^{(n+1)}} = F_{n+1} \mathbf{r} + G_{n+1} \mathbf{v} + H_{n+1} \mathbf{p} \quad (6.10)$$

Equating terms of Eq. (6.9) and Eq. (6.10) gives the recursive coefficient definitions:

$$\begin{aligned} F_{n+1} &= F'_n + cr^\alpha G_n \left(-\frac{\mu}{r^3} \right) \\ G_{n+1} &= G'_n + cr^\alpha F_n \end{aligned} \quad (6.11)$$

$$H_{n+1} = H'_n + cr^\alpha G_n$$

In order to integrate the time with respect to τ , a fourth series is added:

$$\Delta t \simeq \sum_{n=1}^N T_n \frac{\Delta\tau^n}{n!} \quad (6.12)$$

$$T_{n+1} = T'_n \quad (6.13)$$

The coefficients are initialized using $F_0 = 1$, $G_0 = 0$, $H_0 = 0$ and $T_1 = cr^\alpha$, since $dt = cr^\alpha d\tau$, and for $\Delta\tau = 0$, $\mathbf{r} = F_0 \mathbf{r}_0 + G_0 \mathbf{v}_0 + H_0 \mathbf{p} = \mathbf{r}_0$.

When applying the recursion equations, in order to compute the prime quantities of Eq. (6.11), intermediate variables and differentiation rules have to be defined. To demonstrate, the recursion formulas of Eq. (6.11) are applied

as follows, starting with order 1:

$$\begin{aligned}
F_1 &= F'_0 + sG_0 \left(\frac{-\mu}{r^3} \right) = 0 \\
G_1 &= G'_0 + sF_0 = s \\
H_1 &= H'_0 + sG_0 = 0 \\
T_1 &= s
\end{aligned} \tag{6.14}$$

where $s = cr^\alpha$. For order 2:

$$\begin{aligned}
F_2 &= F'_1 + sG_1 \left(\frac{-\mu}{r^3} \right) = s^2 \left(\frac{-\mu}{r^3} \right) \\
G_2 &= G'_1 + sF_1 = s' = \alpha s^2 \frac{\mathbf{r} \cdot \mathbf{v}}{r^2} \\
H_2 &= H'_1 + sG_1 = s^2 \\
T_2 &= T'_1 = s' = \alpha s^2 \frac{\mathbf{r} \cdot \mathbf{v}}{r^2}
\end{aligned} \tag{6.15}$$

The scalar $\mathbf{r} \cdot \mathbf{v}$ is renamed k_1 . In order to get order 3, the recursion equations need to be applied once more, and both k_1 and r will need to be differentiated, which will generate new intermediate variables and differentiation rules. The process is repeated until all the variables and their derivatives with respect to τ are defined in terms of known quantities. The complete set is:

$$r' = cr^\alpha \frac{dr}{dt} = cr^\alpha \frac{r\dot{r}}{r} = cr^\alpha \frac{\mathbf{r} \cdot \mathbf{v}}{r} = s \frac{k_1}{r} \tag{6.16}$$

$$s \equiv cr^\alpha \quad s' = \alpha s^2 \frac{\mathbf{r} \cdot \mathbf{v}}{r^2} = \alpha s^2 \frac{k_1}{r^2} \tag{6.17}$$

$$k_1 \equiv \mathbf{r} \cdot \mathbf{v} \quad k'_1 = s \left(v^2 - \frac{\mu}{r} + \mathbf{p} \cdot \mathbf{r} \right) = s \left(k_4 - \frac{\mu}{r} + k_2 \right) \tag{6.18}$$

$$k_2 \equiv \mathbf{p} \cdot \mathbf{r} \quad k'_2 = s \mathbf{p} \cdot \mathbf{v} = sk_3 \tag{6.19}$$

$$k_3 \equiv \mathbf{p} \cdot \mathbf{v} \quad k'_3 = s \left(\frac{-\mu}{r^3} \mathbf{p} \cdot \mathbf{r} + p^2 \right) = s \left(\frac{-\mu}{r^3} k_2 + p^2 \right) \tag{6.20}$$

$$k_4 \equiv \mathbf{v} \cdot \mathbf{v} \quad k'_4 = 2s \left(\mathbf{p} \cdot \mathbf{v} - \frac{\mu}{r^3} \mathbf{r} \cdot \mathbf{v} \right) = 2s \left(k_3 - \frac{\mu}{r^3} k_1 \right) \tag{6.21}$$

$$\mathbf{p}' = \mathbf{0}, \quad c' = \alpha' = \mu' = 0 \tag{6.22}$$

Using the recursive equations Eq. (6.11) and Eq. (6.13), and the intermediate variables and their derivatives defined in Eqs. (6.16)-(6.22), the F_N , G_N , H_N , and T_N terms can be developed up to any order N , as shown in Algorithm 2. Equation (6.4) is used to obtain the position vector \mathbf{r} . To compute the velocity vector \mathbf{v} , Eq. (6.4) is differentiated with respect to time:

$$\begin{aligned}\mathbf{v} &= \frac{d\mathbf{r}}{dt} = \frac{d\mathbf{r}}{d\tau} \frac{d\tau}{dt} \\ &= \frac{1}{s} \left(\sum_{n=1}^N \left[F_n|_{\tau_0} \left(\frac{\Delta\tau^{(n-1)}}{(n-1)!} \right) \right] \mathbf{r}_0 + \sum_{n=1}^N \left[G_n|_{\tau_0} \left(\frac{\Delta\tau^{(n-1)}}{(n-1)!} \right) \right] \mathbf{v}_0 \right. \\ &\quad \left. + \sum_{n=1}^N \left[H_n|_{\tau_0} \left(\frac{\Delta\tau^{(n-1)}}{(n-1)!} \right) \right] \mathbf{p} \right)\end{aligned}\quad (6.23)$$

Note that this form of the velocity computation corresponds to the τ -velocity implementation presented in Section 2.2.3.2.

The computation of the elapsed time is achieved using Eq. (6.12). Algorithm 2 gives the detailed steps for the computation of the F and G Stark series coefficients. In this study, the implementation of Algorithm 2 is performed with the symbolic manipulator Maple, and the software's code generation capabilities are employed to obtain optimized expressions for the coefficients. Details on the implementation are provided in Appendix C, as well as the coefficients of all four series up to order six, which is the first order for which all the variables introduced in Eqs. (6.16)-(6.22) are used. The Fortran coefficient files as well as an integration routine using the F and G Stark Series can be found at: http://russell.ae.utexas.edu/index_files/fgstark.html and as an electronic supplement to the associated journal publication [252].

An important drawback of the F and G Stark series is the complexity of

the high order coefficients. This complexity, associated with taking repeated analytic derivatives of multivariate functions, leads to large coefficients files when high orders are required. The symbolic manipulator efficiently combats the problem at low orders using intermediate variables to minimize repeated computations (see [Chapter C](#)). However, the complexity burden begins to overwhelm the symbolic manipulator near order 25, and the files become harder to compile; the method becomes inefficient, and the authors decided against computing the series beyond order 25. As a matter of fact, as is shown in [Section 6.3.1](#), this limitation only presents a small inconvenience in practice, as discretizations of 15 or more points per revolution typically require orders below 20.

Algorithm 2 Computation of the coefficients F and G Stark series

- 1: **Input:** $\mathbf{r}_0, \mathbf{v}_0, \mathbf{p}, \mu, \alpha$
 - 2: Initialize $k_1 \leftarrow \mathbf{r} \cdot \mathbf{v}, k_2 \leftarrow \mathbf{p} \cdot \mathbf{r}, k_3 \leftarrow \mathbf{p} \cdot \mathbf{v}, k_4 \leftarrow \mathbf{v} \cdot \mathbf{v}, s \leftarrow cr^\alpha$
 - 3: Initialize $F_0 \leftarrow 1, G_0 \leftarrow 0, H_0 \leftarrow 0$ and $T_1 \leftarrow s$
 - 4: **for** $n \leftarrow 0, N - 1$ **do**
 - 5: **Compute** F'_n, G'_n, H'_n, T'_n (i.e. differentiate each coefficient with respect to τ)
 - 6: **Substitute** $\{r' = s\frac{k_2}{r}, s' = \alpha s^2 \frac{k_1}{r^2}, k'_1 = s(k_4 - \frac{\mu}{r} + k_2), b' = sk_3,$
 $k'_3 = s(-\frac{\mu}{r^3}k_2 + p^2), k'_4 = 2s(k_3 - \frac{\mu}{r^3}k_1)\}$
 - 7: $F_{n+1} \leftarrow F'_n + sG_n(-\frac{\mu}{r^3})$
 - 8: $G_{n+1} \leftarrow G'_n + sF_n$
 - 9: $H_{n+1} \leftarrow H'_n + sG_n$
 - 10: $T_{n+1} \leftarrow T'_n$
 - 11: **end for**
 - 12: **Output:** $\{F_n, G_n, H_n, T_n\}$ for $n = 0 \rightarrow N$
-

6.2.2 F and G CRTBP Series

As for the Stark problem, the authors have studied the development of the series for the CRTBP using a variety of basis vectors, and a variety of formulations of the problem (including the propagation in the rotating frame), but the resulting expressions lead to less convenient sets of recursion formulas. The best option of those considered uses four vectors to span the full space: \mathbf{r}_0 , \mathbf{v}_0 , $\mathbf{r}_{m_1,0}$, and $\mathbf{v}_{m_1,0}$. The space can not be spanned under all circumstances, using any three of the vectors. A fourth vector is potentially redundant, however, is necessary to ensure the recursions.

The position and velocity vectors have to be expressed as Taylor series in the independent variable τ . For order N , the position vector \mathbf{r} at $\tau = \tau_0 + \Delta\tau$ can be written as:

$$\mathbf{r} \simeq \sum_{n=0}^N \left. \frac{d^n \mathbf{r}}{d\tau^n} \right|_{\tau_0} \frac{\Delta\tau^n}{n!} \quad (6.24)$$

Since the set of four vectors \mathbf{r}_0 , \mathbf{v}_0 , $\mathbf{r}_{m_1,0}$, $\mathbf{v}_{m_1,0}$ forms a basis of the 3D space, the consecutive derivatives of r in Eq. (6.24) can be expressed in this basis:

$$\left. \frac{d^n \mathbf{r}}{d\tau^n} \right|_{\tau_0} = F_n|_{\tau_0} \mathbf{r}_0 + G_n|_{\tau_0} \mathbf{v}_0 + A_n|_{\tau_0} \mathbf{r}_{m_1,0} + B_n|_{\tau_0} \mathbf{v}_{m_1,0} \quad (6.25)$$

Equation (6.25) is plugged in Eq. (6.24):

$$\begin{aligned} \mathbf{r} &\simeq \sum_{n=0}^N \left[F_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \mathbf{r}_0 + \sum_{n=0}^N \left[G_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \mathbf{v}_0 \\ &\quad + \sum_{n=0}^N \left[A_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \mathbf{r}_{m_1,0} + \sum_{n=0}^N \left[B_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \mathbf{v}_{m_1,0} \\ &\equiv F\mathbf{r}_0 + G\mathbf{v}_0 + A\mathbf{r}_{m_1,0} + B\mathbf{v}_{m_1,0} \end{aligned} \quad (6.26)$$

The extended Lagrange coefficients form of the solution is:

$$\mathbf{r}(\tau = \tau_0 + \Delta\tau) = f\mathbf{r}_0 + g\mathbf{v}_0 + a\mathbf{r}_{m_1,0} + b\mathbf{v}_{m_1,0} \quad (6.27)$$

Therefore, taking the decomposition in Eq. (6.26) to infinity, and comparing it term by term to Eq. (6.27) yields a relation between the f , g , a , b functions and their associated series:

$$\begin{aligned} f &= \sum_{n=0}^{\infty} \left[F_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \\ g &= \sum_{n=0}^{\infty} \left[G_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \\ a &= \sum_{n=0}^{\infty} \left[A_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \\ b &= \sum_{n=0}^{\infty} \left[B_n|_{\tau_0} \frac{\Delta\tau^n}{n!} \right] \end{aligned} \quad (6.28)$$

As for the Stark case, in order to obtain the coefficients of the F and G CRTBP series for any order n , the successive derivatives of \mathbf{r} at τ are decomposed in the basis formed by \mathbf{r} , \mathbf{v} , \mathbf{r}_{m_1} and \mathbf{v}_{m_1} . It is important to note that this decomposition, when evaluated at τ_0 , is the one shown in Eq. (6.25).

$$\frac{d^n \mathbf{r}}{d\tau^n} = F_n \mathbf{r} + G_n \mathbf{v} + A_n \mathbf{r}_{m_1} + B_n \mathbf{v}_{m_1} \quad (6.29)$$

Equation (6.29) can be differentiated with respect to τ , using Eq. (2.33) and Eq. (2.49):

$$\begin{aligned} \frac{d^{n+1} \mathbf{r}}{d\tau^{n+1}} &= F'_n \mathbf{r} + F_n s \mathbf{v} + G'_n \mathbf{v} + G_n s(p \mathbf{r} + q \mathbf{r}_{m_1}) \\ &\quad + A'_n \mathbf{r}_{m_1} + A_n s \mathbf{v}_{m_1} + B'_n \mathbf{v}_{m_1} - B_n s \mathbf{r}_{m_1} \\ &= (F'_n + spG_n) \mathbf{r} + (G'_n + sF_n) \mathbf{v} \\ &\quad + (A'_n + sqG_n - sB_n) \mathbf{r}_{m_1} + (B'_n + sA_n) \mathbf{v}_{m_1} \end{aligned} \quad (6.30)$$

Alternatively, Eq. (6.29) can be advanced from n to $n+1$:

$$\frac{d^{n+1} \mathbf{r}}{d\tau^{n+1}} = F_{n+1} \mathbf{r} + G_{n+1} \mathbf{v} + A_{n+1} \mathbf{r}_{m_1} + B_{n+1} \mathbf{v}_{m_1} \quad (6.31)$$

Comparing Eq. (6.30) and Eq. (6.31) yields the recursion formulas for F , G , A , and B :

$$\begin{aligned} F_{n+1} &= F'_n + spG_n \\ G_{n+1} &= G'_n + sF_n \\ A_{n+1} &= A'_n + sqG_n - sB_n \\ B_{n+1} &= B'_n + sA_n \end{aligned} \tag{6.32}$$

As mentioned earlier, a fifth series is necessary in order to integrate the time with respect to τ :

$$\Delta t \simeq \sum_{n=1}^N \frac{d^n t}{d\tau^n} \frac{\Delta\tau^n}{n!} = \sum_{n=1}^N T_n \frac{\Delta\tau^n}{n!} \tag{6.33}$$

$$T_{n+1} = T'_n \tag{6.34}$$

For $\Delta\tau = 0$, $\mathbf{r}(\tau_0) = F_0\mathbf{r}_0 + G_0\mathbf{v}_0 + A_0\mathbf{r}_{m_1,0} + B_0\mathbf{v}_{m_1,0} = \mathbf{r}_0$. The coefficients are therefore initialized using $F_0 = 1$, $G_0 = A_0 = B_0 = 0$. Since $dt/d\tau = s$, $T_0 = 0$ and $T_1 = s$.

When using the recursion equations described in Eq. (6.32) and Eq. (6.34) at order n , the derivatives of the coefficients of previous orders with respect to τ have to be computed. In order to compute those derivatives in a recursion, a set of new intermediate variables and differentiation rules are necessary. As for the Stark case, a complete set requires that derivatives of all the variables with respect to τ can be expressed as explicit functions of the variables. The creation of the set is initiated by the differentiation of r_1 and r_2 , which appear

in p and q , as well as in the Sundman transformation term s .

$$r'_1 = s \frac{\mathbf{r}_1 \cdot \mathbf{v}_1}{r_1} = s \frac{\mathbf{r} \cdot \mathbf{v} - \mathbf{r}_{m_1} \cdot \mathbf{v} - \mathbf{r} \cdot \mathbf{v}_{m_1}}{r_1} = s \frac{k_1 - k_2 - k_3}{r_1} \quad (6.35)$$

$$r'_2 = s \frac{\mathbf{r}_2 \cdot \mathbf{v}_2}{r_2} = s \frac{\mathbf{r} \cdot \mathbf{v} + c\mathbf{r}_{m_1} \cdot \mathbf{v} + c\mathbf{r} \cdot \mathbf{v}_{m_1}}{r_2} = s \frac{k_1 + c(k_2 + k_3)}{r_2} \quad (6.36)$$

$$k_1 \equiv \mathbf{r} \cdot \mathbf{v} \quad k'_1 = s(\mathbf{v} \cdot \mathbf{v} + p\mathbf{r} \cdot \mathbf{r} + q\mathbf{r} \cdot \mathbf{r}_{m_1}) = s(k_4 + pk_5 + qk_6) \quad (6.37)$$

$$\begin{aligned} k_2 \equiv \mathbf{r}_{m_1} \cdot \mathbf{v} \quad k'_2 &= s(\mathbf{v}_{m_1} \cdot \mathbf{v} + p\mathbf{r}_{m_1} \cdot \mathbf{r} + q\mathbf{r}_{m_1} \cdot \mathbf{r}_{m_1}) \\ &= s(k_7 + pk_6 + qr_{m_1}^2) \end{aligned} \quad (6.38)$$

$$k_3 \equiv \mathbf{r} \cdot \mathbf{v}_{m_1} \quad k'_3 = s(\mathbf{v} \cdot \mathbf{v}_{m_1} - \mathbf{r} \cdot \mathbf{r}_{m_1}) = s(k_7 - k_6) \quad (6.39)$$

$$k_4 \equiv \mathbf{v} \cdot \mathbf{v} \quad k'_4 = s(2p\mathbf{v} \cdot \mathbf{r} + 2q\mathbf{v} \cdot \mathbf{r}_{m_1}) = 2s(pk_1 + qk_2) \quad (6.40)$$

$$k_5 \equiv \mathbf{r} \cdot \mathbf{r} \quad k'_5 = 2s\mathbf{r} \cdot \mathbf{v} = 2sk_1 \quad (6.41)$$

$$k_6 \equiv \mathbf{r} \cdot \mathbf{r}_{m_1} \quad k'_6 = s(\mathbf{v} \cdot \mathbf{r}_{m_1} + \mathbf{r} \cdot \mathbf{v}_{m_1}) = s(k_2 + k_3) \quad (6.42)$$

$$k_7 \equiv \mathbf{v} \cdot \mathbf{v}_{m_1} \quad k'_7 = s(p\mathbf{r} \cdot \mathbf{v}_{m_1} - \mathbf{v} \cdot \mathbf{r}_{m_1}) = s(pk_3 - k_2) \quad (6.43)$$

Using the recursive equations Eq. (6.32) and Eq. (6.34) with the intermediate results from Eqs. (6.35)-(6.43), the coefficients of the five F and G CRTBP series can be computed up to arbitrary order N and are evaluated at $\tau = \tau_0$. Then, Eq. (6.26) is used to obtain the position vector \mathbf{r} . Once again, in order to compute the velocity vector \mathbf{v} , Eq. (6.26) is differentiated with respect to time:

$$\begin{aligned} \mathbf{v} &= \frac{d\mathbf{r}}{dt} = \frac{d\mathbf{r}}{d\tau} \frac{d\tau}{dt} = \frac{1}{s} \mathbf{r}' \\ &= \frac{1}{s} \left(\sum_{n=1}^N \left[F_n|_{\tau_0} \frac{\Delta\tau^{n-1}}{(n-1)!} \right] \mathbf{r}_0 + \sum_{n=1}^N \left[G_n|_{\tau_0} \frac{\Delta\tau^{n-1}}{(n-1)!} \right] \mathbf{v}_0 \right. \\ &\quad \left. + \sum_{n=1}^N \left[A_n|_{\tau_0} \frac{\Delta\tau^{n-1}}{(n-1)!} \right] \mathbf{r}_{m_1,0} \sum_{n=1}^N \left[B_n|_{\tau_0} \frac{\Delta\tau^{n-1}}{(n-1)!} \right] \mathbf{v}_{m_1,0} \right) \end{aligned} \quad (6.44)$$

As for the Stark case, the relations presented in this section are implemented via the symbolic manipulation software Maple, and using an algorithm similar to [Algorithm 2](#). The code generation capability of Maple is used to generate optimized Fortran code. Similar to the solution to the Stark problem, the main drawback of this technique is the increasing complexity of the terms of high order. The chain rules involved in the successive differentiations lead to large coefficient files, as the symbolic manipulator and compiler become overwhelmed for high orders. This practical limitation on the order N explains the maximum orders chosen in [Section 6.3.2](#).

6.3 Numerical Validation and Comparison to Other Integrators

The F and G Stark and CRTBP series are applied to a set of test cases and their numerical performance is evaluated, in terms of accuracy and speedup. All simulations in this study use the GNU Fortran compiler (gfortran) v4.7.0 on a Windows 7 workstation with a quad-core Intel Xeon W3550 CPU with 3.07GHz clock speed, and 6GB of RAM. The `-O3` optimization flag was used during compilation. The coefficient files are generated using Maple 17.

6.3.1 F and G Stark Series

This section presents the results obtained when using the F and G Stark series propagator on a set of different test cases. The method is validated and its computational efficiency is compared to both the MTS method

(see Section 6.3.1.1) and the RKF8 numerical integrator. The Sundman transformation makes the local error more uniform and justifies the use of fixed-step integrators. The RKF8 integrator is implemented using both the time-velocity and the τ -velocity equations of motion using Eq. (2.42) and Eq. (2.44), respectively. The Stark time-acceleration is used: $\ddot{\mathbf{r}} = -\frac{\mu}{r^3}\mathbf{r} + \mathbf{p}$ and the gravitational parameter is fixed at $\mu = 1$.

6.3.1.1 The Modern Taylor Series Method

Unlike the F and G approach, Taylor series coefficients for the state variables can also be computed recursively using Leibniz's product rule. The Taylor series formed in this manner is referred to as the Modern Taylor Series (MTS) method and has identical coefficients to those found using the F and G approach. The MTS method was originally implemented for the 2-body problem in the astrodynamics field as the power series method [88]. The method has been shown to be valid for any ODE problem that is piecewise smooth [160], and, as previously mentioned, has been used recently for the integration of a low-thrust trajectory using the classic Sundman transformation [353]. Other recent astrodynamics applications of the MTS method include Scott and Martini [301], where the efficiency of the MTS method is highlighted for near Keplerian orbits. Furthermore, in Jorba and Zou [160], the method is shown to perform similarly to conventional integration methods on a restricted three body problem simulation, making the MTS a useful benchmark in the current study to evaluate the performance of the proposed F and G Stark series solutions. The main challenge in implementing the MTS is splitting

the dynamics into the elementary operations such as additions, subtractions, multiplications, divisions, exponents, etc. This decomposition can be done manually [301] (not advisable as the dynamics become more complicated) or using Automatic Differentiation (AD) packages [160, 37, 339, 16, 132].

For the dynamics of the problem addressed in this project, only multiplication, division, and raising to a power are needed. These operations have to be carried out on Taylor series and therefore, the general Leibniz product rule can be used:

$$[f(t)g(t)]^{(n)} = \sum_{k=0}^n \binom{n}{k} f^{(k)}(t)g^{(n-k)}(t) \quad (6.45)$$

where $x^{(n)}$ represents the n -th derivative of x . Using the general Leibniz rule, relations for other common elementary operations can be derived [160]. All the recursion relations for the current study are implemented in Fortran. Splitting the dynamics into elementary operations is carried out manually for the relatively simple Stark dynamics, and checked with the output produced by *Taylor* [160].

6.3.1.2 Accuracy Estimation

There are many different methods to estimate the accuracy of an integration method, as presented in Berry and Healy [20]. This section uses comparisons with a truth trajectory computed using a variable-step Runge-Kutta-Fehlberg integrator of order (7)8, in quad precision, with a tolerance set to 10^{-25} . However, this section presents two ways of obtaining an estimate of the accuracy without having to propagate the trajectory a second time. The

integrals of motion method should be used with caution ; the highest order terms method can be used in order to implement a variable-step integrator from the F and G solution.

Integrals of Motion Integrals of motion play an important role, when available, in numerical simulations. The integration accuracy can crudely be represented by the variation in any integral of motion, and checking for this conservation constitutes a reasonable accuracy check when no truth or other integration method is available. However, as shown by Huang and Innanen [153], there might be integration errors that are not represented by a change in the integral invariants. The detailed expressions of the Stark problem's integrals of motion can be found in Lantoine and Russell [176]. The Hamiltonian, most commonly used in the Stark case, is simply:

$$H = \frac{1}{2} v^2 - \frac{\mu}{r} - \mathbf{r} \cdot \mathbf{p} \quad (6.46)$$

and the associated error estimation is:

$$\epsilon_H = |H(\mathbf{r}, \mathbf{v}, \mathbf{p}) - H(\mathbf{r}_0, \mathbf{v}_0, \mathbf{p})| \quad (6.47)$$

where r and v are the series solutions computed from Eq. (6.4) and (6.23) respectively, and $|x|$ is the absolute value of the dummy scalar x .

Magnitude of Highest Order Terms The preservation of the Hamiltonian can be a good indication of the accuracy of the Taylor series solution. However, the change in the Hamiltonian across a single step can not be computed to a value below machine epsilon (10^{-16} for double precision, 10^{-32} for

quad precision), since the computation subtracts terms of large magnitude (v^2 , μ , r). Another way to estimate the accuracy of the solution is to compute the contribution of the last term added to the series. This method is similar to the classic accuracy checks made in most variable-step integrators. As opposed to the method using the Hamiltonian, the last term error estimates are not based on a difference of terms of large magnitude, and can therefore retain values as low as 10^{-307} for double precision (10^{-4931} for quad), which are the underflow limits of the used hardware and compiler. Such information is useful for the design of a variable-order or variable-step integration routine. If the F and G Stark series are computed up to order N , then the error estimation terms are defined as follows:

$$\epsilon_R = \left\| (F_N|_{\tau_0} \mathbf{r}_0 + G_N|_{\tau_0} \mathbf{v}_0 + H_N|_{\tau_0} \mathbf{p}) \frac{\Delta\tau^N}{N!} \right\| \quad (6.48)$$

$$\epsilon_V = \left\| \frac{1}{s} (F_N|_{\tau_0} \mathbf{r}_0 + G_N|_{\tau_0} \mathbf{v}_0 + H_N|_{\tau_0} \mathbf{p}) \frac{\Delta\tau^{N-1}}{(N-1)!} \right\| \quad (6.49)$$

$$\epsilon_T = \left| T_N \frac{\Delta\tau^N}{N!} \right| \quad (6.50)$$

where $\|\mathbf{x}\|$ is the Euclidian norm of the dummy vector \mathbf{x} .

Overflow Safeguard When numerically computing the F and G series, it is important to detect rapid divergence to avoid the potential for arithmetic overflow. In the context of an automated use of the method (in an optimizer for instance), the step size has to be controlled, since Taylor series have a convergence region; a step taken outside of that region could lead to divergence, and therefore arithmetic overflow. There are several ways to avoid arithmetic overflow, even without formally detecting divergence. One simple heuristic is

to compute the ratio described in [Eq. \(6.51\)](#) for each series and for each order n . If the ratio exceeds some user-defined threshold (e.g. 100), the algorithm returns with an error flag. This safeguard is based on the idea that in a converging Taylor Series, the magnitude of high-order terms is smaller than that of lower-order terms. The ratio is defined as such:

$$\gamma_F(n) = \frac{\left| (F_n|_{\tau_0} \frac{\Delta\tau^n}{n!}) \right|}{1 + \left| (F_i|_{\tau_0} \frac{\Delta\tau^i}{i!}) \right|} \quad (6.51)$$

where i is the order of the first non-zero term in the series.

6.3.1.3 Study of the Generalized Sundman Transformation for the Unperturbed 2-Body Problem

Many authors have studied the effects of the Sundman transformation on the performance of integrators. As mentioned in [Section 2.2.3](#), the benefits of the Sundman transformation include a reduction and an equalization of the local error, allowing fixed-step integrators to perform similarly to their variable-step counterparts. In this section, the local truncation error is analyzed for the various power laws associated with the Sundman transformation. Unlike conventional integration methods, the Taylor series techniques are shown to be most efficient for $\alpha = 1$.

In this study, the local error is represented by the change in Hamiltonian over one integration step. The orbit used for the testing is an unperturbed Keplerian orbit with semi-major axis $a = 1$ LU (LU = arbitrary Length Unit) and eccentricity $e = 0.6$, propagated for one full period with 50 equally spaced points in τ . The propagation starts at perapse. The Taylor series methods

are used with order 8 and 12, which result in fairly large changes in the Hamiltonians; indeed, it is important for this test that the round-off error can be neglected with respect to the local truncation error. [Figure 6.1](#) shows the change in Hamiltonian across a single orbit for different α values. [Figure 6.2](#) shows the maximum change in the Hamiltonian and the Root Mean Square (RMS) of this change over one period versus α for all integrators. The Taylor series solutions are presented with two different orders to show how the error is translated when the order changes. The plots of the total error and of the local error standard deviation are not shown because they exhibit similar behavior to that of [Fig. 6.2](#). As mentioned, the results presented on [Fig. 6.1](#) and [Fig. 6.2](#) are obtained when propagating an unperturbed Kepler orbit. A similar study of the error has been conducted when adding a Stark acceleration. The behavior of the local error is similar to that of the unperturbed problem, so the non-zero perturbation plots are not included here.

For the case of the Runge-Kutta integrator, the results presented in [Fig. 6.1c](#) closely agree with past findings (see [Section 2.2.3.1](#)). The local error depends on the position of the particle along the orbit, and this behavior itself depends on α in the Sundman transformation. When $\alpha = 0$ (the independent variable is equivalent to time), the traditional rise of the error around periape is observed, as well as a low error around apoapse. As α grows, the local error becomes more uniform, until the trend reverses near $\alpha = 2$. [Figure 6.2](#) confirms this behavior, and shows that the ideal α for the presented case is between $3/2$ and 2 , leading to minimized maximum numerical deviation of the Hamiltonian and minimized RMS of that deviation. When using the τ -velocity equations

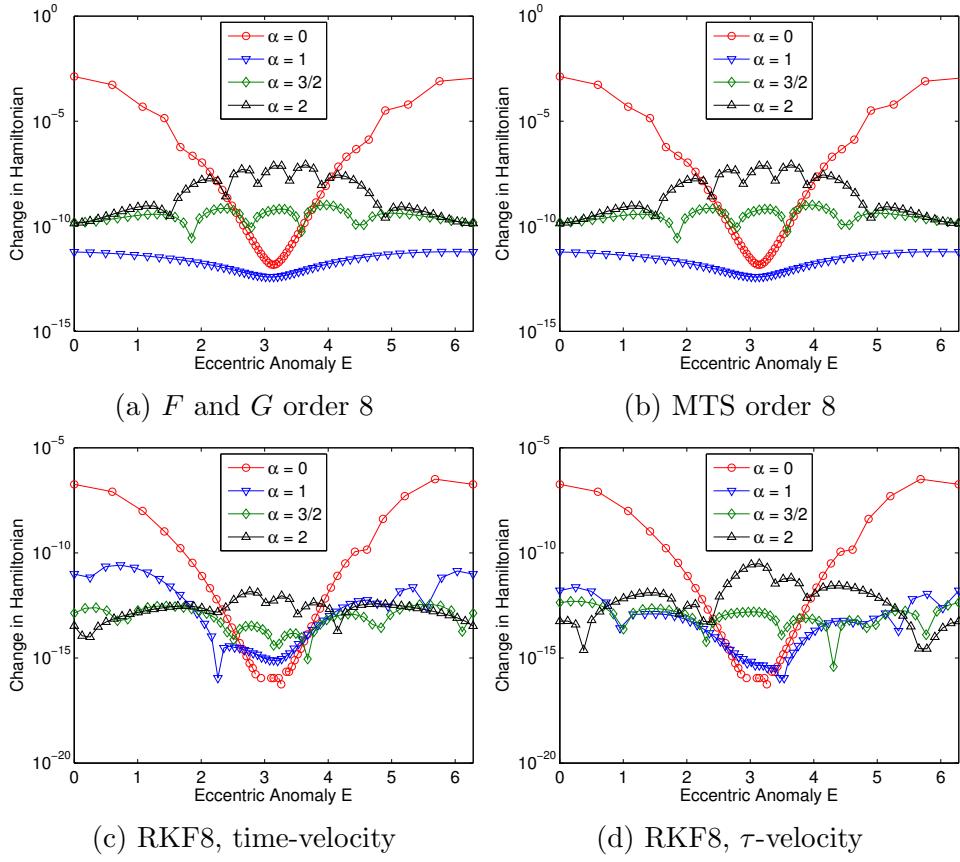


Figure 6.1: Effect of orbit location on the local truncation error. $a = 1$,
 $e = 0.6$, $\mathbf{p} = \mathbf{0}$, $\mu = 1$

of motion and the RKF8 integrator, the ideal α is considerably lower (see Fig. 6.2). Figure 6.1d shows that the local error is more uniform for $\alpha = 1$ when using the τ -velocity formulation compared to using the time-velocity formulation.

The results obtained when using either of the two Taylor series integrators demonstrate a peculiar behavior. As illustrated in Fig. 6.2, there is a narrow, but real, dip of the maximum and RMS errors at $\alpha = 1$. This be-

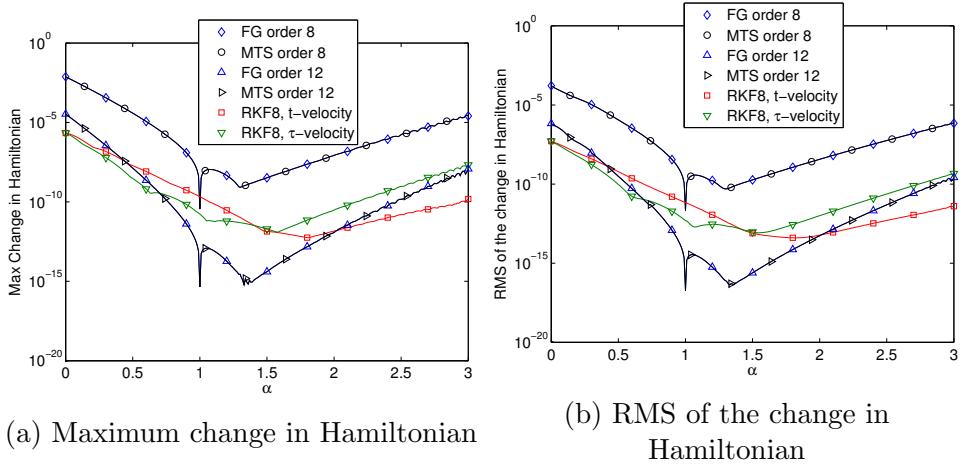


Figure 6.2: Effect of α on the local truncation error. α equally spaced using $\Delta\alpha = 0.01$, $a = 1$, $e = 0.6$, $\mathbf{p} = \mathbf{0}$, $\mu = 1$

havior is observed for both the F and G method and the MTS method, and is confirmed by Fig. 6.1a and Fig. 6.1b. The local error when $\alpha = 1$ is lower and more uniform than for any other value of α . Aside from the sharp dip at $\alpha = 1$, the error statistics for other values of α behave similarly to their RKF8 counterparts. The optimal α for α not close to 1 is $\alpha \approx 1.25$, a further shift to the left of the RKF8 integrations. It is speculated that the narrow dip at $\alpha = 1$ is in part due to the recursion equations Eq. (6.35)eq:auxend which are highly simplified for the exact case of $\alpha = 1$. This peculiar behavior presents a serendipitous advantage: the $\alpha = 1$ case corresponds to the regular geometric distribution of nodes along an ellipse, and is ideal for the preliminary low-thrust optimization problem and other problems benefiting from a regular discretization. Therefore, in the case of the Taylor series methods, the ideal α for discretization coincides with the ideal α for accuracy and efficiency.

6.3.1.4 Runtime Comparisons Between the Taylor Series and the RKF8 Integrator

Here, the F and G integrator is compared to the MTS integrator and to both the time- and τ -velocity implementations of the RKF8 integrator. The methods are compared on the basis of CPU time necessary for the propagation of approximately one revolution of a perturbed orbit.

In order to compare the speed of different integrators in a fair manner, it is important to ensure that the accuracy obtained with those integrators is comparable. In this section, the accuracy ϵ is defined as the norm of the difference between the truth and the state vector at the final time of the integrator being tested, as shown in [Eq. \(6.52\)](#). The truth is obtained using a validated variable-step Runge-Kutta-Fehlberg integrator of order (7)8, in quad precision. The tolerance of the integration is set to 10^{-25} .

$$\epsilon = \|\mathbf{X}(\tau_f)_{truth} - \mathbf{X}(\tau_f)\| \quad (6.52)$$

The comparison scheme relies on the computation of n , the minimum number of steps necessary to achieve a prescribed accuracy. This dynamic selection of n ensures that the accuracy is reached, while minimizing the computational efforts for each integrator being tested. The comparisons, made according to [Algorithm 3](#), allow for a fair evaluation of the timings: the accuracy and perturbation discretization achieved by each integration method are essentially the same.

In order to present a complete set of test cases, the procedure described in [Algorithm 3](#) is repeated for a variety of parameters and initial conditions.

Algorithm 3 Comparison of the timings for all integrators

```
1: Initialize the propagation (choose  $a, e, \mathbf{p}, \alpha, \tau_f, N, \epsilon$ )
2: Compute the truth using a variable-step Runge-Kutta-Fehlberg integrator
   and quad precision.
3: Compute the minimum number of steps  $n_{RK}$  necessary to achieve the toler-
   ance  $\epsilon$  with RKF8, using either the  $\tau$ -velocity or time-velocity formulation.
4: for  $k \leftarrow 1, N$  do
5:   Compute the minimum number of steps  $n_k$  required to achieve  $\epsilon$  using
   each of the TS methods and order  $k$ 
6:   for  $i \leftarrow 1, 10000$  do
7:     Propagate the orbit for  $n_k$  steps, using the corresponding TS method
     and time the execution using the CPU_TIME Fortran procedure
8:   end for
9:   Average the timings
10:  Compute  $n_{st} = \text{CEILING}(n_{RK}/n_k)$ . The RKF8 propagation is done over
     $n_k$  segments of  $n_{st}$  steps each, simulating a classic discretization of the
    orbit in  $n_k$  segments
11:  for  $i \leftarrow 1, 10000$  do
12:    Propagate the orbit using RKF8 and time the execution using
    CPU_TIME procedure
13:  end for
14:  Average the timings
15: end for
```

The nominal trajectory is an ellipse specified by $a = 1$ LU and $e = 0.8$, and is in the plane of the first two coordinates. The nominal perturbation is $\mathbf{p} = 10^{-3}[1, 1, 1]^T$ LU/TU² (TU = arbitrary Time Unit) and the nominal tolerance is $\epsilon = 10^{-12}$. [Figure 6.3](#) presents the timings obtained using all integration methods while varying the tolerance, the eccentricity, the magnitude of the perturbation and the Sundman transformation power law.

[Figure 6.3](#) shows that the Taylor series solutions are significantly faster than the RKF8 integration in most cases. It also shows that the τ -velocity solution is faster than its time-velocity counterpart, for reasons previously

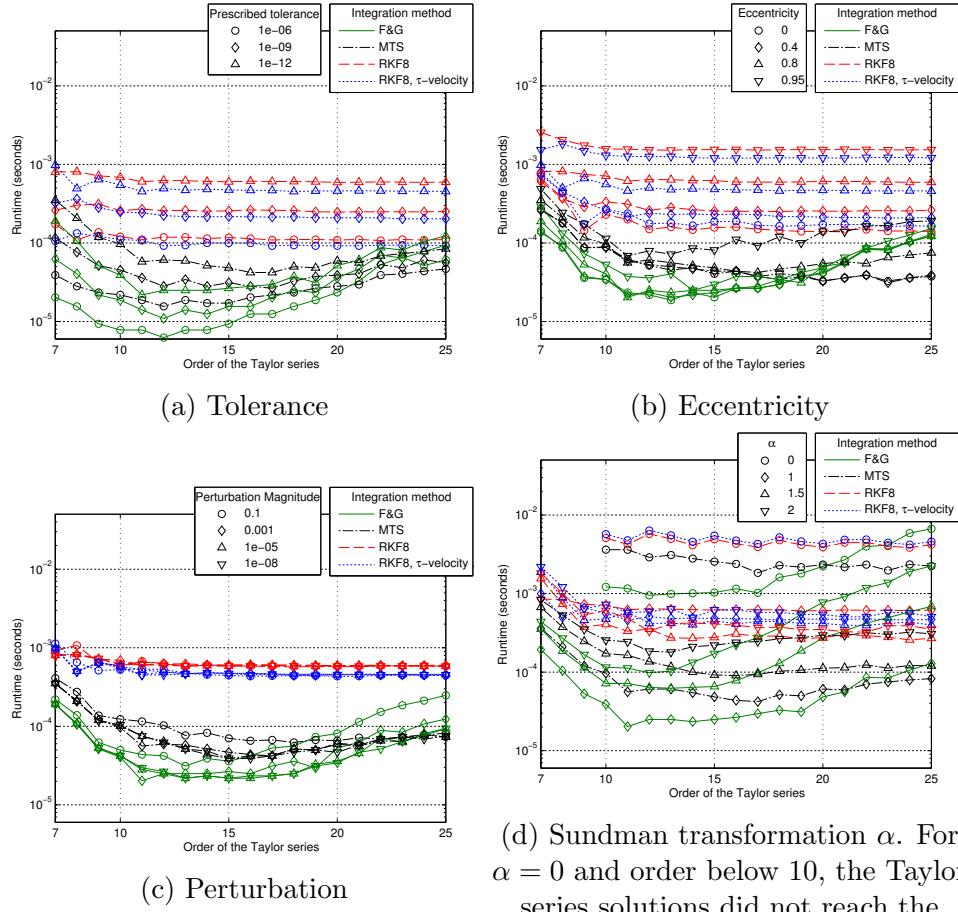


Figure 6.3: Timing results when varying integration parameters, for all integrators

described. The timings of the RKF8 are approximately constant over any TS order. The increase in runtime for the RKF8 integrator at the low TS orders occurs because the integrator cannot take less than 1 step per segment. Therefore in these regions, a lower order Runge-Kutta integrator would be necessary to reduce runtime without over-achieving the accuracy. The F and G series demonstrate optimal efficiency for orders between 10 and 15. The

MTS method becomes more efficient than the F and G method beyond order 20.

The F and G series are increasingly inefficient at high orders due to the complexity mentioned in [Section 6.2.1](#), making their coefficients difficult to obtain beyond order 25. Alternatively, it is noted that the MTS method easily achieves orders of 100 or more, although numerical issues have been observed to accumulate when taking very large steps, lowering the interest of using the method at a very high order.

[Figure 6.3a](#) shows that the prescribed tolerance, while having a significant influence on the timings, does not modify the relative performance of the methods. It is observed in [Fig. 6.3b](#) that the eccentricity, while significantly affecting the timings of the Runge-Kutta method, leaves the timings of the Taylor series methods essentially unchanged, and therefore has an important impact on the relative performances of the integrators. The speedups of the F and G Stark series solution over the RKF8 integrator go from $5\times$ for $e = 0$ up to $60\times$ for $e = 0.95$. [Figure 6.3c](#) shows that the magnitude of the perturbation does not alter the performance of the RKF8 integrations. However, it does have a minor impact on the timings of the Taylor series solutions, and therefore on the relative performances. A large perturbation tends to counteract the benefits of using the Taylor series. Finally, [Fig. 6.3d](#) demonstrates the influence of the Sundman transformation power law. The runtimes to achieve the prescribed accuracy for all methods are higher when $\alpha = 0$ (corresponding to no transformation). As expected from the results in [Fig. 6.1](#) and [Fig. 6.2](#), the F and G series performs best at $\alpha = 1$ for the majority of orders.

The primary advantage of the F and G method lies in the fact that the intermediate orders for which it performs optimally correspond to a practical range of node points per revolution. As an example, to achieve a prescribed relative accuracy of 10^{-12} , an F and G series solution of order 12 requires approximately 20 points per revolution. This approximate number of node points and the geometrically symmetric distribution associated with $\alpha = 1$ makes the F and G method an ideal candidate to replace traditional numerical integrators in applications such as low thrust optimization.

It is emphasized that the favorable results of the TS methods are in part due to two beneficial behaviors. First, the formulation presented in [Section 6.2.1](#) is simple enough to generate relatively small coefficient files. The simple nature of the 2-body equations of motion and the use of an inertially constant perturbation \mathbf{p} as the third basis vector greatly influences the efficiency of the method. It is noted that the authors successfully adapted this F and G series method to other more complicated dynamical systems (e.g. the restricted 3-body problem, and the 2-body problem with a perturbation constant in a rotating frame). However, the timings obtained for those scenarios did not present as significant an improvement when compared to the RKF8 timings. Second, the behavior of the local error for the Taylor series solutions coupled with the Sundman transformation (see [Section 6.3.1.3](#)) is serendipitous, and highly beneficial to the results. Indeed, when the independent variable is proportional to the eccentric anomaly, the Taylor series methods abnormally require fewer steps to achieve equivalent accuracies when compared to all other techniques considered.

6.3.1.5 Application: Low-Thrust Orbital Transfer

In order to show the utility of the F and G series method, an application to a low-thrust trajectory is presented. The problem consists of a transfer from a circular Low-Earth Orbit (LEO) to a highly-elliptical orbit. To simplify the implementation and to avoid Keplerian arcs, the selected strategy involves thrusting constantly. In order to propagate the trajectory using the Stark formulation, a discretization in a finite number of segments with constant perturbation is necessary. The thrust is directed along the velocity vector at the beginning of each segment, and its magnitude varies according to [Eq. \(6.53\)](#).

For segment i , from t_i to t_{i+1} :

$$\mathbf{p}_i = \frac{q}{r(t_i)} [1 + \cos \nu(t_i)] \mathbf{v}(t_i) \quad (6.53)$$

where ν is the true anomaly and q is a constant ($q = 10^{-3}$ in the following).

The control policy and $\Delta\tau$ are computed at each step of the propagation, until a final condition is reached, as described in [Algorithm 4](#) in Appendix B. The test is designed to compare total compute speeds of the Taylor series solutions to the speeds of the RKF8 integrations, while preserving a similar accuracy for any two methods being compared.

[Figure 6.4a](#) presents the trajectory resulting from the control policy of [Eq. \(6.53\)](#) with a discretization of 20 Segments Per Revolution (SPR). The eccentricity increases from 0 to 0.94, with large changes when the particle is close to periapse (when the thrust is the largest). The size of the τ -steps has a similar behavior. [Figure 6.4b](#) shows the variations in the magnitude of the thrust. In this application, the accuracy is once again represented by the

magnitude of the difference between the last propagated state and the last state of the truth. The truth is computed using a quad precision variable-step Runge-Kutta-Fehlberg integrator of order (7)8.

[Figure 6.5](#) and [Fig. 6.6](#) present the results in terms of 1) achieved speedup and 2) order of magnitude of the common accuracy, as functions of the SPR discretization. The goal of displaying the data in this manner is to emulate realistic trajectory design applications. The desired accuracy and associated model discretization typically have large effects on the number of optimization variables, and therefore are of primary interest to a trajectory designer. [Figure 6.5](#) and [Fig. 6.6](#) provide the expected speedups with respect to these two important design variables. Each curve in [Fig. 6.5](#) and [Fig. 6.6](#) stops when the speedup for a larger SPR is significantly smaller than that for a smaller SPR. As a matter of fact, a large decrease of the speedup means that the accuracy has reached machine epsilon for this order, and increasing the SPR will only increase the runtime, not the precision. The confirmation of this behavior can be observed on the accuracy plots of [Fig. 6.5](#) and [Fig. 6.6](#).

[Figure 6.5a](#) shows that, for this example, the F and G Stark series method provides speedups between $4\times$ and $15\times$ when compared to the time-velocity RKF8. For orders 7 and 10, the speedups are relatively constant across all discretizations, and the accuracy is low, as expected. For high orders (> 15), the accuracy is high, but the speedups are lower, confirming the tendency observed in [Section 6.3.1.4](#). The best results for the F and G method are obtained for intermediate orders (between 10 and 15), and for discretizations with $\text{SPR} > 20$. A low SPR results in lower accuracies for the TS methods,

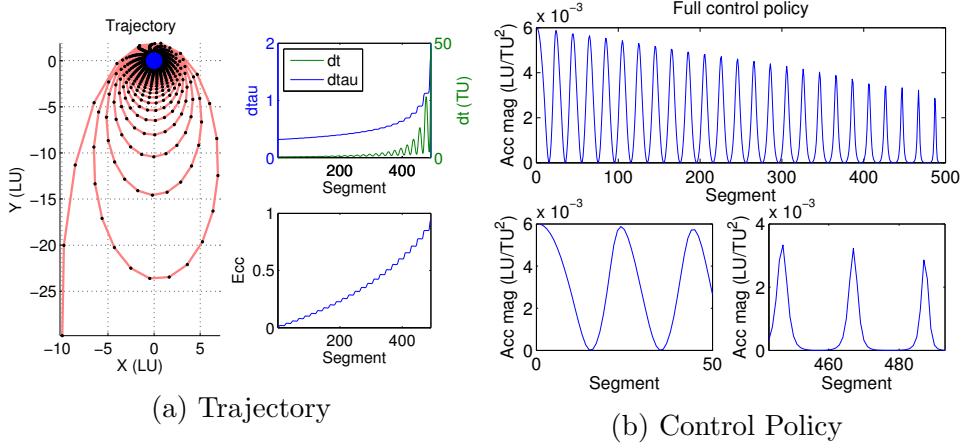


Figure 6.4: Application: low-thrust orbital transfer

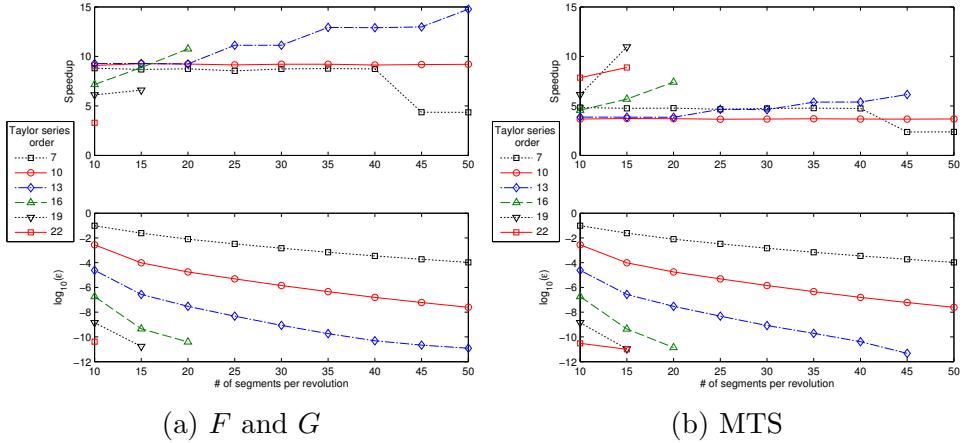


Figure 6.5: Results of the application: Speedup of Taylor series solutions compared to time-velocity RKF8

and since the RKF8 does not have to take many steps per segments to match these accuracies, the associated speedups are low.

[Figure 6.5b](#) shows the results obtained when comparing the MTS method to the RKF8 integrations. On a general level, the speedups are lower, confirming the findings of [Section 6.3.1.4](#).

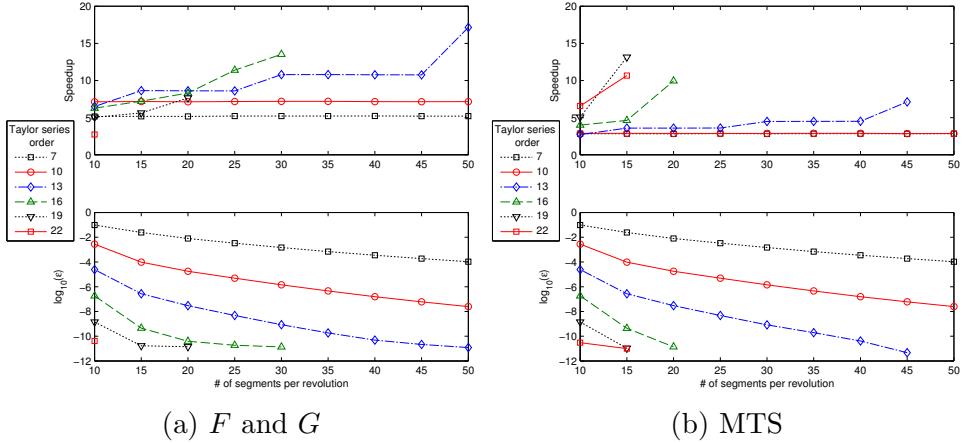


Figure 6.6: Results of the application: Speedups and accuracy of Taylor series solutions compared to τ -velocity RKF8

Figure 6.6 presents the results obtained when comparing the Taylor series solutions to the τ -velocity RKF8. As expected, the behaviors are identical, with lower speedups due to the better efficiency of the τ -velocity RKF8.

6.3.2 F and G CRTBP Series

This section presents the results of the Taylor series integration of the CRTBP for different scenarios. The method is validated, and its performance is compared to that of the same RKF8 integrator as in Section 6.3.1. Both variable-step and fixed-step integrations are considered and compared. The Sundman type transformations are used for the fixed-step integration, and the behavior of the associated solutions is studied. The maximum series orders are: 15 with no Sundman transformation, 13 in the $s = r_1$ and $s = r_2$ cases, and 12 in the $s = r_1 r_2$ case. All timing results are sensitive to implementation

choices and compilation settings; however, a good-faith effort was made by the authors to ensure fair comparisons of the integration methods. The timings are obtained using the intrinsic Fortran routine `cpu_time`, and averaging over 10,000 simulations. Four different scenarios have been considered in this study, in order to qualitatively capture the different types of motion in the CRTBP. The scenarios correspond to four different periodic orbits, each propagated for one period.

6.3.2.1 Framework

The CRTBP studied in this section is a representation of the Earth-Moon system, with the mass ratio $\mu = 1/82.27 = 0.012155099064057$ given by Broucke [49]. The initial conditions for the primary are:

$$[\mathbf{r}_0 \ \mathbf{v}_0]_{m_1}^T = [-\mu \text{ LU}, 0 \text{ LU}, 0 \text{ LU}, 0 \text{ LU/TU}, -\mu \text{ LU/TU}, 0 \text{ LU/TU}]^T \quad (6.54)$$

and the secondary initial conditions are given by Eq. (2.23) and Eq. (2.24).

This section presents the four different scenarios studied for the analysis of the F and G CRTBP series. The first three scenarios are planar periodic orbits, with initial conditions found in Broucke [49], and the fourth one is a 3D Halo orbit found using differential correction. In Scenario 1, the particle orbits the primary m_1 . In Scenario 2, the particle stays close to m_2 . In Scenario 3, the particle goes back and forth between m_2 and m_1 . Table 6.1 contains the initial conditions for each orbit, as well as their period, and the distance of their closest approach to each mass. Figure 6.7 to Fig. 6.10, in addition to presenting the studied trajectories, show the influence of the Sundman

Table 6.1: Initial conditions for the four test scenarios

	Scenario 1	Scenario 2	Scenario 3	Scenario 4
x_0 (LU)	-1.972795736	1.044045197	0.960920244	0.974785881
y_0 (LU)	0.0	0.0	0.0	0.0
z_0 (LU)	0.0	0.0	0.0	0.071295151
\dot{x}_0 (LU/TU)	0.0	0.0	0.0	0.0
\dot{y}_0 (LU/TU)	1.864677641	-0.633046910	-0.958491618	-0.526306975
\dot{z}_0 (LU/TU)	0.0	0.0	0.0	0.0
T_P (TU)	6.283185420	6.283159790	18.489999998	2.517727406
m_1 dist. (LU)	0.022365	0.83006	0.36744	0.88259
m_2 dist. (LU)	0.59037	0.039515	0.026908	0.072481

Scenario 1 is Orbit 33 on Table 3 of Broucke [49]

Scenario 2 is Orbit 65 on Table 9 of Broucke [49]

Scenario 3 is Orbit 263 on Table 5 of Broucke [49]

Scenario 4 is obtained using a differential corrector and well known techniques for finding Halo orbits

Earth radius $R_E = 0.01659$ LU ; Moon radius $R_M = 0.004519$ LU

transformations in each case. Time is effectively “slowed down” when the particle approaches one of the masses, allowing for a more robust integration, in particular for a fixed-step integrator. Section 6.3.2.3 confirms the utility of the Sundman type transformations. Figure 6.7 to Fig. 6.10 are obtained using the fixed-step twelfth order F and G solution, with the minimum number of steps required so that $\epsilon_{\text{glob}} \leq 10^{-6}$ (see Section 6.3.2.2 and Section 6.3.2.3). Not all integration steps are shown, and the spacing between shown integration steps is displayed on the plot. Figure 6.10 is a plot of the 3D Halo trajectory, and of the projections of this 3D trajectory on the (x, y) , (y, z) , and (x, z) planes.

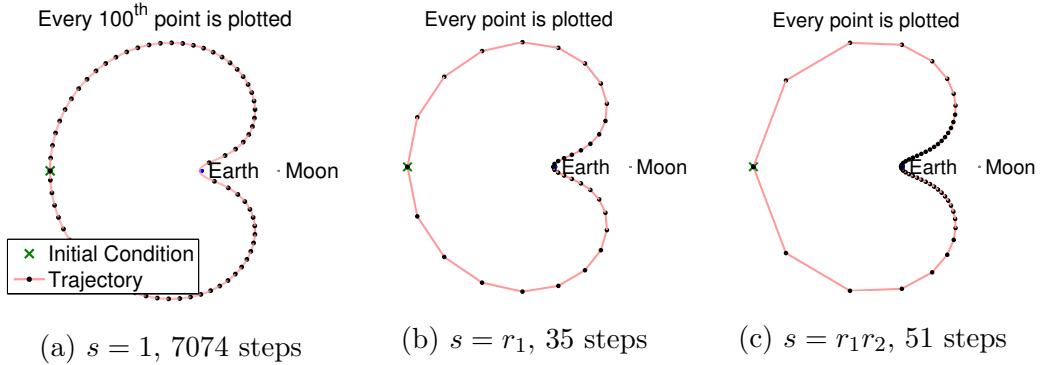


Figure 6.7: Scenario 1, with different Sundman transformations.

$$\epsilon_{\text{glob}} \leq 10^{-6}$$

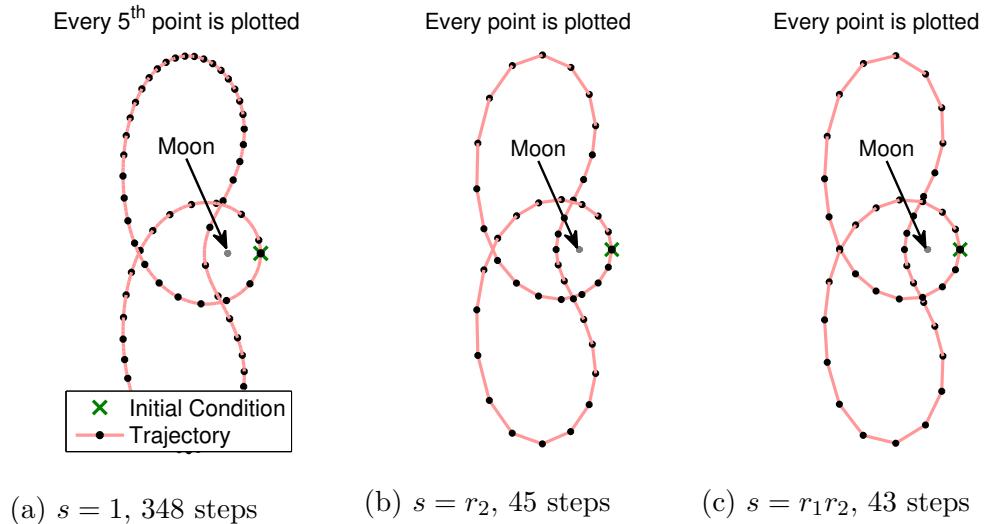


Figure 6.8: Scenario 2, with different Sundman transformations.

$$\epsilon_{\text{glob}} \leq 10^{-6}$$

6.3.2.2 Variable-Step Integration

The four scenarios are integrated using a variable-step integration method.

The F and G CRTBP series solution is compared to a conventional RKF(7)8.

In the case of the series solution, the error estimate, necessary to adjust the step-size, is given by the infinity norm of the last vector added to the series.

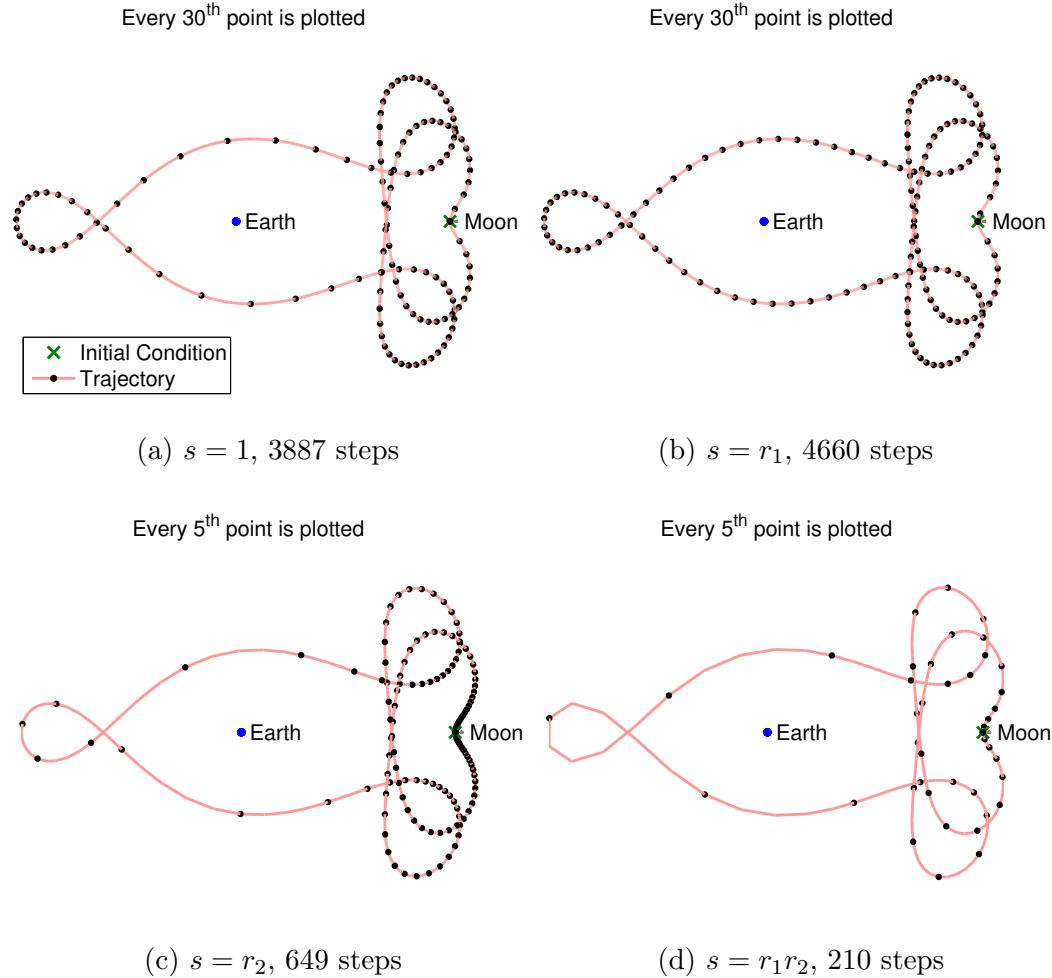


Figure 6.9: Scenario 3, with different Sundman transformations.

$$\epsilon_{\text{glob}} \leq 10^{-6}$$

For order N , the estimate of the error is:

$$\epsilon = \|[\epsilon_r^T \quad \epsilon_v^T \quad \epsilon_t^T]^T\|_\infty \quad (6.55)$$

$$\epsilon_r = (F_N|_{\tau_0} \mathbf{r}_0 + G_N|_{\tau_0} \mathbf{v}_0 + A_N|_{\tau_0} \mathbf{r}_{m_1,0} + B_N|_{\tau_0} \mathbf{v}_{m_1,0}) \frac{\Delta\tau^N}{N!} \quad (6.56)$$

$$\epsilon_v = \frac{1}{s} (F_N|_{\tau_0} \mathbf{r}_0 + G_N|_{\tau_0} \mathbf{v}_0 + A_N|_{\tau_0} \mathbf{r}_{m_1,0} + B_N|_{\tau_0} \mathbf{v}_{m_1,0}) \frac{\Delta\tau^{N-1}}{(N-1)!} \quad (6.57)$$

$$\epsilon_t = T_N|_{\tau_0} \frac{\Delta\tau^N}{N!} \quad (6.58)$$

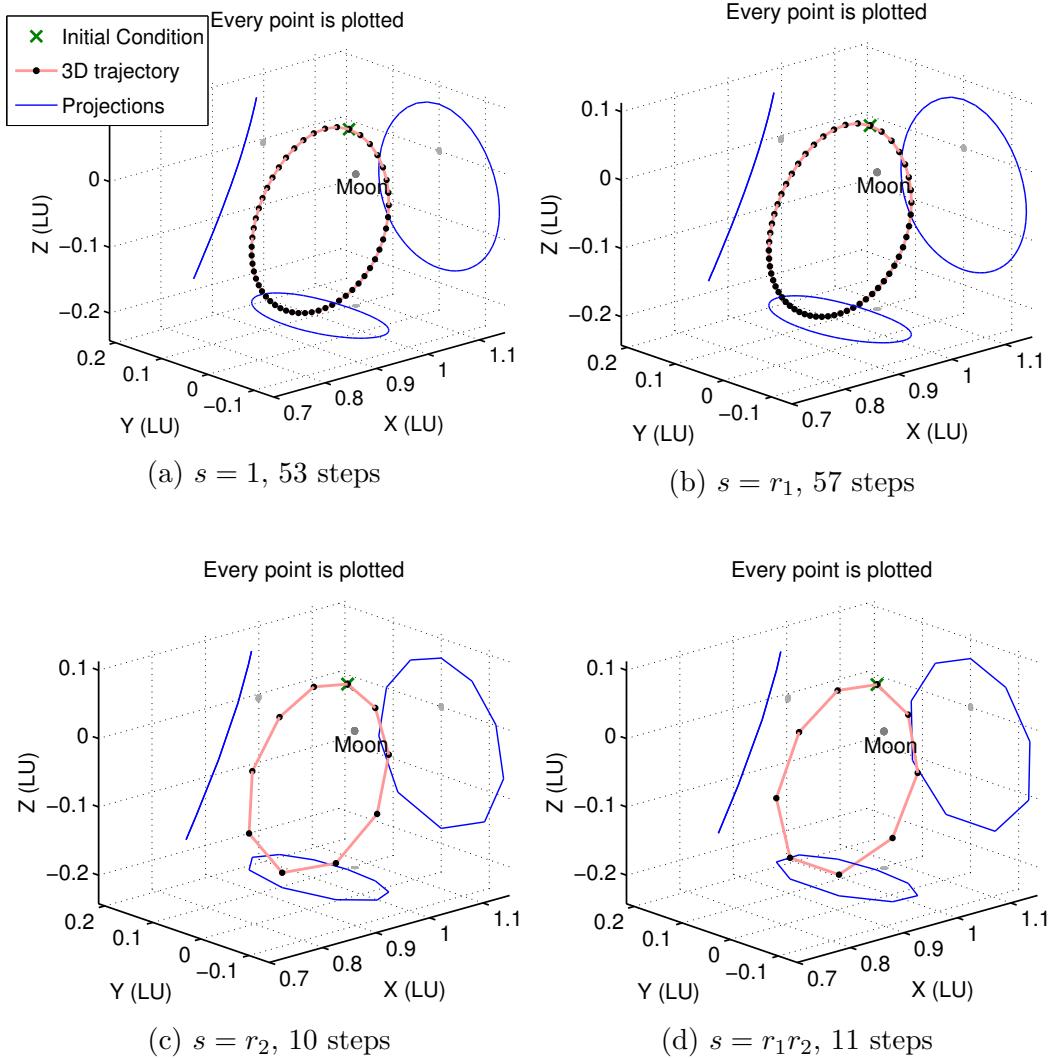


Figure 6.10: Scenario 4, with different Sundman transformations.

$$\epsilon_{\text{glob}} \leq 10^{-6}$$

This error estimate is used in the variable-step integrator in a manner similar to the Runge-Kutta variable-step integrators described by Press et al. [267]. The user inputs an absolute (ϵ_{abs}) and a relative (ϵ_{rel}) tolerance, and at each

step the algorithm ensures that:

$$\epsilon \leq \max(\epsilon_{\text{abs}}, \epsilon_{\text{rel}} \|\mathbf{X}\|_{\infty}) \quad (6.59)$$

In the following, the tolerances are set such that $\epsilon_{\text{abs}} = \epsilon_{\text{rel}}$, and only ϵ_{abs} is mentioned.

In order to compare the time required by different integration routines, it is important to ensure that similar accuracies are obtained in each case. The global accuracy ϵ_{glob} is defined as the norm of the difference between the truth and the state vector at the final time of integration, as shown in [Eq. \(6.60\)](#). The truth is computed using a variable-step RKF(7)8 in quad precision, with tolerance $\epsilon_{\text{abs}} = 10^{-22}$.

$$\epsilon_{\text{glob}} = \|\mathbf{X}(\tau_f)_{\text{TRUTH}} - \mathbf{X}(\tau_f)\| \quad (6.60)$$

To ensure a fair comparison of methods, the prescribed error tolerance ϵ_{abs} for each integrator is successively reduced, until the accuracy ϵ_{glob} described in [Eq. \(6.60\)](#) steps below a prescribed error threshold (ϵ_{thresh}). This comparison scheme not only controls the local error (an inherent feature of the variable-step algorithms), it also ensures that the global errors accumulated over one period are comparable. Moreover, slowly reducing ϵ_{abs} ensures that ϵ_{thresh} is reached while minimizing the computational efforts.

[Figure 6.11](#) presents the speedups obtained when comparing the variable-step F and G CRTBP solution to the variable-step RKF(7)8 for each scenario. No time transformation is used for the variable-step propagation ($s = 1$). The results are given for different error thresholds (ϵ_{thresh}). Missing data points

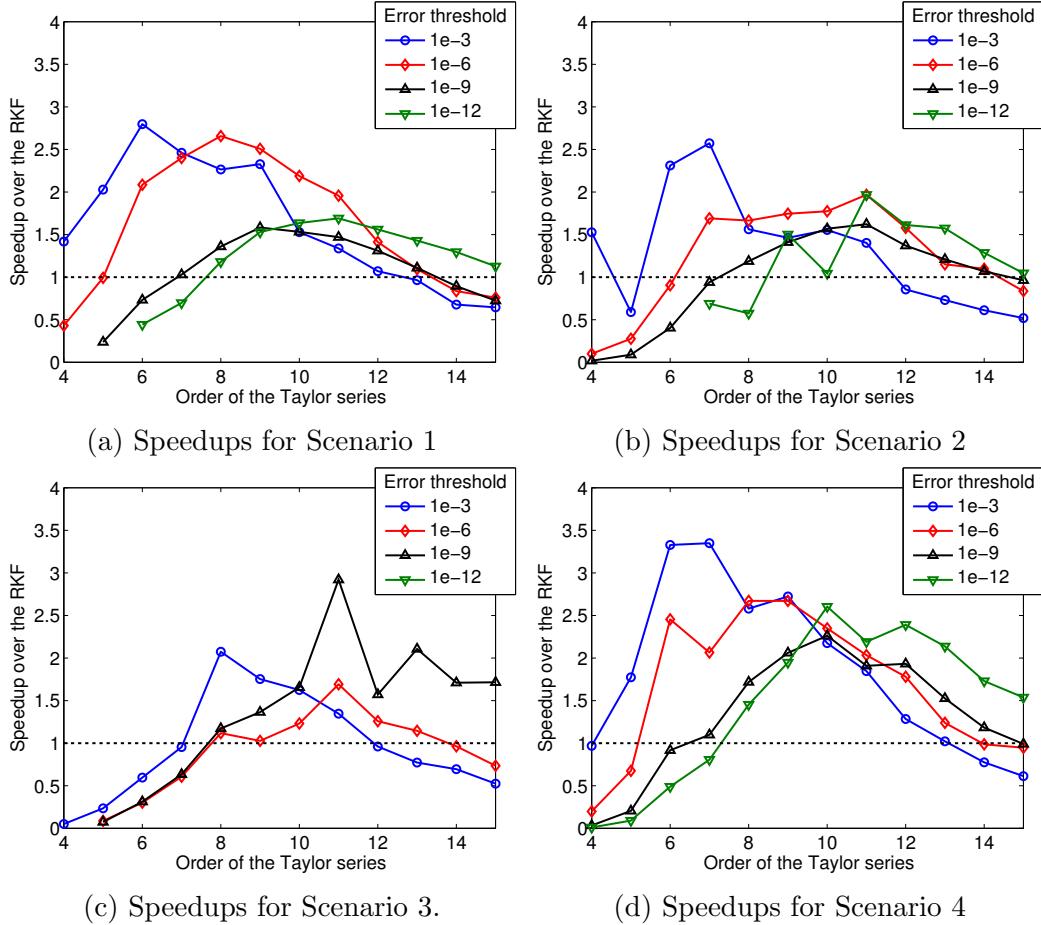


Figure 6.11: Speedups of the variable-step F and G CRTBP propagator compared to the RKF(7)8. No Sundman transformation is used ($s = 1$).

For Scenario 3, for $\epsilon_{thresh} = 10^{-12}$, neither the F and G series nor the RKF converged

correspond to cases where the lower-order F and G series did not converge even for the lowest possible value of ϵ_{abs} ($\epsilon_{abs} = 10^{-16}$).

The behavior of the variable-step F and G integration presented in Fig. 6.11 is similar for all four scenarios. The F and G Taylor series solution is up to $3.3 \times$ faster than the RKF(7)8. Moreover, for low-fidelity propaga-

tions ($\epsilon_{thresh} = 10^{-3}$), the F and G series are between two and three times faster than the RKF(7)8 for all test cases, when selecting an appropriate order. In general, as the prescribed error threshold is lowered, the optimal order increases. The results of Fig. 6.11 are consistent with previously published comparisons between power series solutions of the CRTBP and Runge-Kutta integration: Jorba and Zou [160] demonstrate that their power series solution to the CRTBP performs up to three times faster than a Runge-Kutta integration.

6.3.2.3 Fixed-Step Integration

Comparisons of the fixed-step integrators are performed under conditions similar to the variable-step comparisons. The accuracy is the one described in Eq. (6.60), and the comparison scheme also relies on bringing this accuracy below a prescribed error threshold ϵ_{thresh} . However, in order to reach ϵ_{thresh} , the number of steps taken by the algorithm is slowly increased instead of lowering the tolerance ϵ_{abs} .

Figure 6.12 presents the speedups obtained when comparing the fixed-step F and G CRTBP solution to the fixed-step RKF8. The results are given for different ϵ_{thresh} and for the different Sundman transformations presented in Section 2.2.3.3. Figure 6.13 shows the number of steps necessary to achieve the prescribed error threshold $\epsilon_{thresh} = 10^{-6}$ for each method, and for the different Sundman transformations. Two different orders of the F and G solutions are presented.

Figure 6.12 exhibits a behavior of the fixed-step F and G integra-

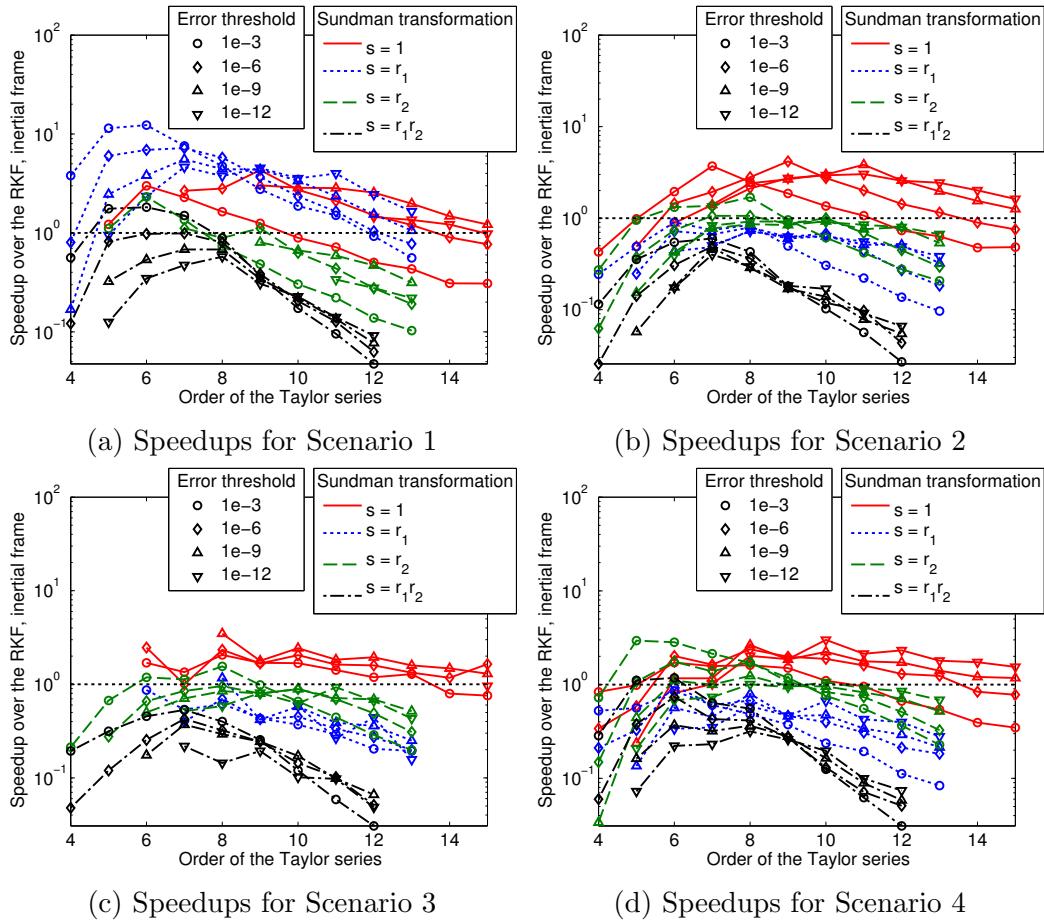


Figure 6.12: Speedups of the fixed-step F and G CRTBP propagator compared to the RKF8

tion similar to that of the variable-step integration. The Taylor series solution presents a different optimal order for each error threshold, increasing as the threshold is lowered. The speedups over the RKF8 go as high as $13\times$. The Sundman transformation usually lowers the speedups, indicating that the RKF8 benefits most from the regularization. This change in relative performance when using a Sundman transformation is due to the increase in the complexity of the equations of motion. This affects the F and G recursion

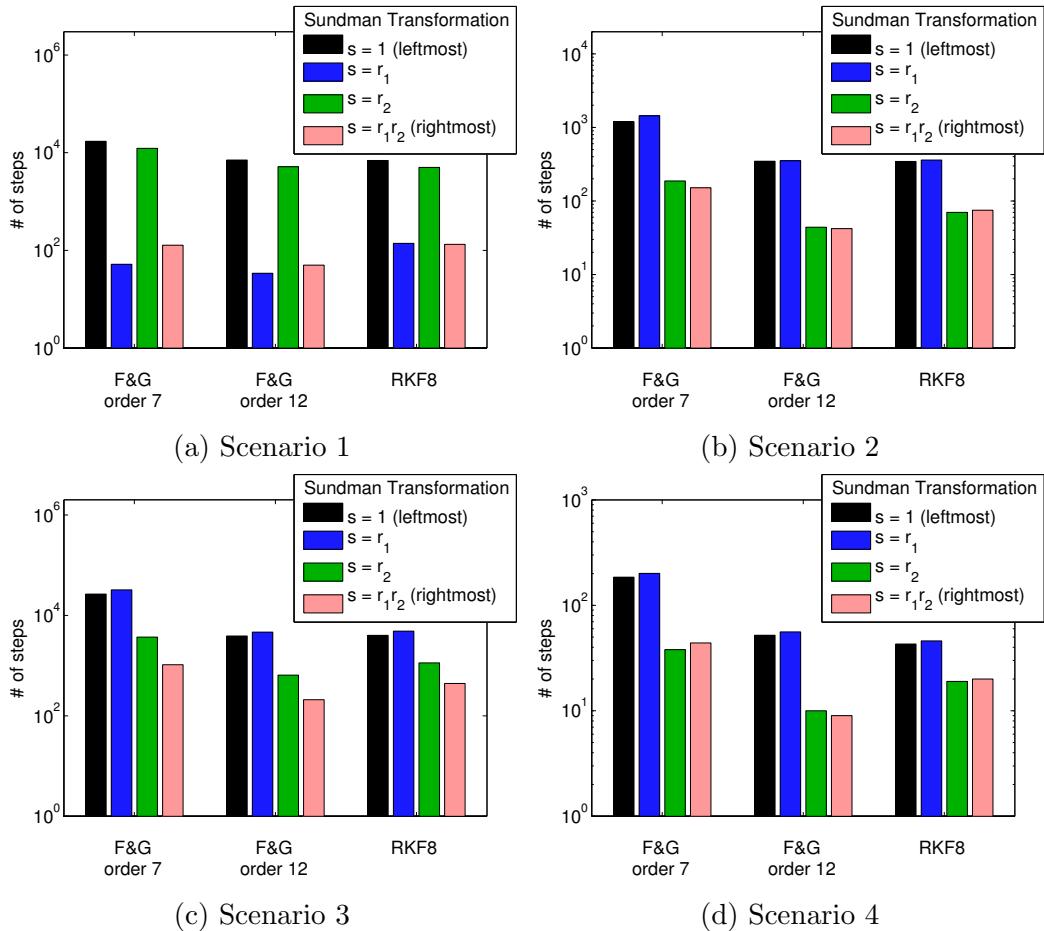


Figure 6.13: Number of steps necessary to achieve the prescribed tolerance $\epsilon_{\text{thresh}} = 10^{-6}$ using fixed-step integration.

complexity more than that of the Runge-Kutta equations. Both methods benefit from the improved discretizations.

Figure 6.13 confirms these tendencies: the number of necessary integration steps is greatly reduced when using an adapted Sundman transformation, for every integration method. As expected based on the proximity of the trajectories to each primary, the best transformations are $s = r_1$ for Scenario 1,

$s = r_2$ for Scenarios 2 and 4, and $s = r_1r_2$ for Scenario 3. However, in all cases the difference between the ideal transformation and the $s = r_1r_2$ is relatively small, making the $s = r_1r_2$ transformation the most useful across all cases. Up to two orders of magnitude fewer steps are necessary to converge when using the $s = r_1r_2$ transformation, compared to the time propagation. As noted, the effect of the regularization on the minimum number of steps are comparable for each method; the resulting speedups are explained by the much increased complexity of the coefficient files of the Taylor series.

6.4 Conclusions

In this chapter, the Lagrange coefficients and their associated F and G Taylor series are derived for the Stark, Kepler and CRTB problems. The Stark effect can be used to approximate a variety of astrodynamics systems, such as low-fidelity low-thrust trajectories or slowly varying perturbations, and the CRTBP can be used in trajectory design to take advantage of multi-body dynamics. A regularization based on the Sundman transformation is used to allow for powerful discretization schemes, decreasing the need for a variable-step integrator. Exact recursion formulas for the obtention of the coefficients of the Taylor series are presented.

It is shown that the Sundman exponent has great influence on the discretization and the equalization of the local error. For the Kepler and Stark problems, the Taylor series methods exhibit a peculiar behavior for the unity power law, which corresponds to making the independent variable proportional to the eccentric anomaly, and results in a regular geometric discretization.

This case is proven to enhance the accuracy of both Taylor series methods, and therefore their computational efficiencies. This result is a significant finding, since the Sundman transformation is conventionally assumed to perform best with the $3/2$ power law. This peculiar behavior of the Taylor series solutions make them ideal candidates for use in applications such as preliminary low-thrust optimization, where regular geometric discretizations are naturally preferred.

When propagating the CRTBP, the “hybrid” Sundman transformation, regularizing the equations of motion at both primaries, is shown to have the most satisfactory results across all test cases. It permits a reduction of the number of necessary integration steps over the time propagation, for all test cases, and for all methods. However, the regularizations increase the complexity of the Taylor series recursion formulae, therefore slowing down the F and G method. On the other hand, the RKF performs particularly well under the transformations, suggesting the utility of fixed-step RKF integrators using Sundman transformed CRTBP equations of motion. Such an implementation is not common in practice, but the analysis in this study gives evidence of its benefit.

The F and G Stark series integration method is shown to perform between 3 and 60 times faster than a conventional RKF8 integration scheme, where performance is tied mostly to eccentricity, order of the Taylor series, and the exponent in the Sundman transformation. The new F and G Stark series method proves most efficient for orders between 10 and 15, corresponding to discretizations of approximately 20-40 points per revolution. High ec-

centricities are shown to increase the relative efficiency of the method, while large perturbations tend to diminish the efficiency. The Modern Taylor Series method is used as a second benchmark to compare computational efficiency. The MTS method is shown to be slower for low to intermediate orders, but becomes more efficient than the F and G Stark series method for orders above 20. It is noted that the particular cases of the Kepler and Stark problems are especially well suited for the Taylor series solutions, since they produce moderately complex recursion equations. The F and G CRTBP series are validated on four different test cases designed to represent different orbit classes of the CRTBP, in both 2D and 3D. The overall performance of the F and G Taylor series solutions is comparable to that of the conventional numerical integrator, when considering all the parameters above. The results are consistent with a previous study comparing a power series integration method to an RKF8, implying that the F and G method is qualitatively similar in performance to other power series methods. In addition, the variable-step F and G series with no time transformation are demonstrated to perform between two and three times faster than the conventional RKF(7)8 for low-fidelity propagations, making them particularly useful for preliminary design of three-body trajectories.

Chapter 7

Conclusions

In the context of longer and more complex space exploration missions, the computational efficiency and robustness of optimization algorithms are paramount. This dissertation presents advances to the state-of-the-art of optimal control solvers, as well as numerical propagation and partial computation methods, which are combined to improve the efficiency of numerical trajectory optimization techniques. A summary of the contributions was given in [Section 1.4](#) and a list of the publications related to the present research can be found in [Appendix D](#). The present section summarizes the main conclusions and explores avenues for future work.

The first multiple-shooting formulation of a Differential Dynamic Programming algorithm is developed in [Chapter 3](#). The algorithm uses State-Transition Matrices for sensitivity propagation, a null-space trust-region method for the minimization of quadratic subproblems subject to simple bounds, and an Augmented Lagrangian approach for equality and inequality path and terminal constraints. Moreover, the Symmetric Rank-1 quasi-Newton update can be used to estimate the second-order information at some or all iterations. [Chapter 4](#) presents a set of application cases and their use in the validation and performance analysis of the new algorithm. The quadratic ex-

pansions and update equations of [Chapter 3](#) are validated on quadratic test cases. The treatment of the terminal and path constraints is verified on a number of examples taken from the optimal control and spacecraft trajectory optimization literature. The multiple-shooting formulation is demonstrated to be effective in solving general optimal control problems. The sensitivities of the problems are split when using multiple legs, which should result in an increased robustness to initial conditions. As expected, the multiple-shooting formulation does not perform as well as the single-shooting algorithm when applied to simple optimization problems, which is explained by the increase in number of unknown parameters, and in complexity with the addition of linkage constraints. By definition, problems with low complexity do not benefit from splitting their sensitivities. However, they could still benefit from the multiple-shooting formulation once implemented in parallel. The quasi-Newton approach is demonstrated to be highly beneficial in terms of runtime, yielding speedups up to $10\times$. The decrease in robustness when using approximations of the second-order information can be mitigated when using a re-initialization strategy, and computing full second-order partials every n iterations. While the quasi-Newton approach has less impacts on problems with a high number of legs, it is likely to consistently improve the runtimes of problems with complex equations of motion.

The computation of first- and/or second-order sensitivities is essential to most optimization methods, including the MDDP algorithm. The three most routinely used methods are described and compared in [Chapter 5](#). The propagation of variational equations, finite differences, and complex step

derivatives are compared for both fixed- and variable-step integration. The use of variable-step integration methods is associated with three important subtleties which can potentially deteriorate the partial computation: 1) when using the variational equations, the propagation error should be computed using the state only; 2) variable-step integration of the variational equation introduces an error due to the sensitivity of the final state to the variation of the step-size. Complex step and finite difference methods are not sensitive to this problem, as they incorporate the effects of step-size variation; 3) variable-step integration can lead to discontinuities in the number of integration steps which irreparably damage the quality of the partials for all methods. Because of these caveats, fixed-step integration associated to time-regularization methods are recommended. Moreover, the complex step derivatives are shown to be faster than finite-difference methods, and only slightly slower than the variational equations. The flexibility offered by the complex step derivatives, as well as their potential for parallel implementation, make them an ideal choice for a general optimal control solver. The implementation of CSD in MDDP increases the versatility of the algorithm, permitting changes in the dynamics, cost and constraint functions with minimal user involvement.

A new propagation method of the Kepler, Stark, and Circular Restricted Three-Body Problems is detailed in [Chapter 6](#), which is based on an extension of the well-known f and g Lagrange coefficients, and their series approximation. The F and G Stark series are demonstrated to significantly outperform both conventional explicit integration and another series representation, especially at high eccentricity, and for moderate accuracy. On the

other hand, the benefits of using the F and G CRTBP series are not as significant, as the method only outperforms traditional integrators for very specific regimes. The effects of time regularizations based on the Sundman transformation are investigated for both classes of problems. The Stark and Kepler series solutions benefit greatly from the regularization, in particular when the independent variable is taken to be proportional to the eccentric anomaly. This result is particularly adapted to the optimization of spacecraft trajectories since the regularization results in a regular geometric discretization, which is adapted to the approximation of continuous thrust with piecewise constant controls. A “hybrid” Sundman transformation regularizing time at both primaries is shown to enhance the discretization of the CRTBP. The conventional fixed-step explicit Runge-Kutta integrator used in the study performs particularly well under the transformation, and their combined use is recommended for three-body trajectory design.

Future Work The analysis of the multiple-shooting algorithm presented in this dissertation should be extended to include problems of higher complexity. The optimization of multiple-body trajectories, utilizing the multiple-phase (and multiple dynamics) capability as well as the sensitivity reduction properties of multiple-shooting, constitutes an important application case for which the MDDP algorithm seems well suited. The overall robustness of the algorithm should be analyzed, using randomly generated initial conditions and algorithm parameters. Finally, the multiple-shooting formulation offers a high potential for parallel implementation, since part of the inner loop problem can

be solved for all legs independently. An extensive study of the benefits of the parallel implementation in terms of runtime is necessary in order to fully evaluate the performance of the MDDP algorithm.

The treatment of inequality constraints, while proved to be effective in its current state, is likely to benefit from smoothing techniques to eliminate the second-order discontinuities. Polynomial and Heaviside approximations are good candidates for smoothing functions, and further investigations of their use are needed.

The current MDDP algorithm relies heavily on optimization algorithms implemented by the author, and in particular on the null-space trust-region method presented in [Section 3.4.1](#). The investigation of the use of different nonlinear programming algorithms for both the inner and outer loops constitutes an interesting avenue for future work. A linearly constrained optimization method should be used when solving for the optimal controls, so that linear constraints are treated within the inner loop, an approach demonstrated to benefit Augmented Lagrangian based algorithms [72]. Small scale SQP methods are ideal candidates for this nonlinear problem. The dimensions of the initial conditions optimization problem suggest that the use of a dedicated large-scale NLP solver such as SNOPT or IPOPT could highly benefit the MDDP implementation, especially in terms of runtime, when utilizing a large number of subintervals. In addition, algorithms exploiting the sparsity of the problem could allow for a second-order update of the path constraints multipliers similar to that of [Section 3.3.3](#).

Finally, while spacecraft trajectory optimization is the focus of most applications in this dissertation, the multiple-shooting formulation of DDP should be beneficial to many other optimal control applications. The implementation developed for this doctoral research is designed to be modular, and to solve the general optimal control problem. Therefore, the investigation of the performance of the MDDP algorithm for different applications, such as robot trajectory optimization problems, constitutes a logical and exciting direction for future studies.

Appendices

Appendix A

Orbital Period in τ for Different Sundman Transformations

For all cases, Kepler's equation is used:

$$M = E - e \sin E = nt \quad (\text{A.1})$$

where $n = \sqrt{\frac{mu}{a^3}}$ is the mean motion, M is the mean anomaly, E is the eccentric anomaly, e is the eccentricity and a is the semi-major axis. Equation (A.1) is differentiated:

$$dM = n dt = (1 - e \cos E) dE \quad (\text{A.2})$$

Using the generalized Sundman transformation:

$$dM = ncr^\alpha d\tau = (1 - e \cos E) dE \quad (\text{A.3})$$

The geometry of the ellipse is also utilized:

$$r = a(1 - e \cos E) \quad (\text{A.4})$$

A.1 Case $d\tau = cdt$

This case is trivial, as the independent variable is proportional to time. Kepler's equation yields:

$$dM = n \ dt = nc \ d\tau \quad (\text{A.5})$$

$$nc \int_0^{\tau_p} d\tau = \int_0^{2\pi} dM \quad (\text{A.6})$$

$$(A.7)$$

Finally:

$$\tau_p = \frac{2\pi}{nc} \quad (\text{A.8})$$

A.2 Case $d\tau = crdt$

Using Eq. (A.3) and Eq. (A.4) for this Sundman transformation:

$$nca(1 - e \cos E) \ d\tau = (1 - e \cos E) \ dE \quad (\text{A.9})$$

$$nca \ d\tau = dE \quad (\text{A.10})$$

This equation shows that the independent variable for the case $\alpha = 1$ is proportional to the eccentric anomaly. Moreover, integrating over one period:

$$nca \int_0^{\tau_p} d\tau = \int_0^{2\pi} dE \quad (\text{A.11})$$

Finally:

$$\tau_p = \frac{2\pi}{nca} \quad (\text{A.12})$$

A.3 Case $d\tau = cr^2 dt$

Using Eq. (A.3) for $\alpha = 2$:

$$ncr^2 d\tau = (1 - e \cos E) dE \quad (\text{A.13})$$

In order to eliminate the r^2 term, and to show that the independent variable is proportional to the true anomaly, a change of variable from dE to $d\nu$ is necessary. Starting from the geometric relations found in Bate et al. [6]:

$$\cos E = \frac{e + \cos \nu}{1 + e \cos \nu} \quad (\text{A.14})$$

$$\sin E = \frac{\sqrt{1 - e^2} \sin \nu}{1 + e \cos \nu} \quad (\text{A.15})$$

Differentiating Eq. (A.14) yields:

$$-\sin E dE = \left[\frac{-\sin \nu}{1 + e \cos \nu} - \frac{(e + \cos \nu)(-e \sin \nu)}{(1 + e \cos \nu)^2} \right] d\nu = \frac{-\sin \nu(1 - e^2)}{(1 + e \cos \nu)^2} d\nu \quad (\text{A.16})$$

Dividing by $\sin E$ and plugging Eq. (A.15) in yields:

$$dE = \frac{\sin \nu(1 - e^2)}{\frac{\sqrt{1 - e^2} \sin \nu}{1 + e \cos \nu}(1 + e \cos \nu)^2} d\nu = \frac{\sqrt{1 - e^2}}{1 + e \cos \nu} d\nu = \sqrt{1 - e^2} \frac{r}{p} d\nu \quad (\text{A.17})$$

Equation (A.17) can be plugged into Eq. (A.13):

$$ncr^2 d\tau = \sqrt{1 - e^2} \frac{r}{p} (1 - e \cos E) d\nu \quad (\text{A.18})$$

Using Eq. (A.4):

$$ncr^2 d\tau = \sqrt{1 - e^2} \frac{r}{p} \frac{r}{a} d\nu \quad (\text{A.19})$$

which yields:

$$d\tau = \frac{\sqrt{1 - e^2}}{panc} d\nu = \frac{\sqrt{1 - e^2}}{a(1 - e^2)anc} d\nu = \frac{1}{\sqrt{pa^{3/2} nc}} d\nu = \frac{1}{\sqrt{\mu pc}} d\nu \quad (\text{A.20})$$

Equation (A.20) shows that the independent variable is proportional to the true anomaly in the case $\alpha = 2$. Moreover, integrating this equation over one period yields:

$$\int_0^{\tau_p} d\tau = \frac{1}{\sqrt{\mu pc}} \int_0^{2\pi} d\nu \quad (\text{A.21})$$

Finally:

$$\tau_p = \frac{2\pi}{\sqrt{\mu pc}} \quad (\text{A.22})$$

A.4 Case $d\tau = cr^{3/2}dt$

For the case $\alpha = 3/2$, or “intermediate anomaly case”, more details are provided by Nacozy [227]. Using Eq. (A.3) for $\alpha = 3/2$:

$$ncr^{3/2} d\tau = dE(1 - e \cos E) \quad (\text{A.23})$$

Combined with Eq. (A.4):

$$d\tau = \frac{dE(1 - e \cos E)}{ca^{3/2}(1 - e \cos E)^{3/2}} \quad (\text{A.24})$$

Integrating the last relation over one period gives:

$$\int_0^{\tau_p} d\tau = \frac{1}{ca^{3/2}n} \int_0^{2\pi} \frac{dE}{\sqrt{1 - e \cos E}} = \frac{1}{c\sqrt{\mu}} \int_0^{2\pi} \frac{dE}{\sqrt{1 - e \cos E}} \quad (\text{A.25})$$

This integration is detailed in Nacozy [227]. The result is:

$$\tau_p = \frac{1}{c\sqrt{\mu}} \frac{4 K\left(\sqrt{\frac{2e}{1+e}}\right)}{\sqrt{1+e}} \quad (\text{A.26})$$

where $K(x)$ is the complete elliptic integral of the first kind with the dummy argument x .

Appendix B

Algorithm Used for the Performance Comparison of all Integrators Applied to an Example Low-Thrust Orbital Transfer

Algorithm 4 Application of all integrators to a low-thrust orbital transfer

```
1: for SPR  $\leftarrow 5, 50$  do (SPR: Segments Per Revolution)
2:   For the Taylor series:
3:   for order  $\leftarrow 3, 25$  do
4:     Call propagateTrajectory and store the control policy and  $\Delta\tau$  histories and accuracy. The propagation stops when the final condition is reached, and the necessary number of segments  $M$  is stored.
5:     for  $i \leftarrow 1, 10000$  do
6:       Propagate trajectory using the precomputed control and  $\Delta\tau$  histories  $\{\mathbf{p}_k, \Delta\tau_k\}$  for  $k = 0, M$ 
7:     end for
8:     Average the timings
9:   end for
10:  For the RKF8: Duplicate (3-10) replacing order from line 3 by SPS (Steps Per Segment) and using the RKF8 in propagateTrajectory
11:  Comparison of the two integration methods for similar accuracy:
12:  for order  $\leftarrow 3, 25$  do
13:    Read accuracy of the Taylor series solution
14:    Select the RKF8 solution that has similar accuracy (characterized by SPS)
15:    Compute speedup
16:  end for
17: end for

18: procedure PROPAGATETRAJECTORY
19:   Input:  $\mathbf{X}_0$ , SPR, order  $N$  or SPS,  $r_{max}$ ,  $\mu$ 
20:    $\mathbf{X}$  is the full state vector (position, velocity, time)
21:   Initialize  $k \leftarrow 0$ ,  $\epsilon \leftarrow 0$ 
22:   repeat
23:     Compute  $\tau_p$  using Table 2.1 and  $\mathbf{X}_k$ 
24:      $\Delta\tau_k \leftarrow \tau_p/SPR$ 
25:     Compute  $\mathbf{p}_k$  using Eq. (6.53) and  $\mathbf{X}_k$ 
26:     Compute Hamiltonian  $H_k$  for  $\mathbf{X}_k$ ,  $\mathbf{p}_k$  using Eq. (6.46)
27:     Propagate  $\mathbf{X}_k(\tau)$  to  $\mathbf{X}_{k+1}(\tau + \Delta\tau)$  with  $\mathbf{p}_k$  for order  $N$  (Taylor series solution) or for SPS (RKF8 solution)
28:     Compute Hamiltonian  $H_{k+1}$  for  $\mathbf{X}_{k+1}$ ,  $\mathbf{p}_k$ 
29:      $\epsilon \leftarrow \epsilon + (H_{k+1} - H_k)^2$ 
30:      $k \leftarrow k + 1$ 
31:   until  $\|\mathbf{r}\| \geq r_{max}$ 
32:    $K \leftarrow k - 1$ 
33:    $\epsilon \leftarrow \epsilon/K$ 
34:   Output:  $\mathbf{X}_f$ ,  $\epsilon$ ,  $K$ ,  $\{\mathbf{p}_k, \Delta\tau_k\}^{231}$  for  $k = 0, K$ 
35: end procedure
```

Appendix C

Implementing the F and G Stark Series with a Symbolic Manipulator

The coefficients of the F and G series are computed using the equations in [Section 6.2.1](#) and implemented using the symbolic manipulator Maple 15.01. The `optimize` option of the code generation and the `simplify(.,'size')` command are used to minimize file sizes and the runtime of the resulting code. It is important to note that the order of the F and G method, in theory unbounded, is practically limited by the size of the corresponding coefficients file, and the ability for a compiler to handle large files efficiently (the maximum size being compiler- and hardware-specific). The Fortran source code of the coefficients computation can be downloaded from: http://russell.ae.utexas.edu/index_files/fgstark.htm¹. The size of the files ranges between 4 and 8MB. The coefficients and associated intermediate variables up to order 6 are shown below.

¹Last accessed March 21 2017

$$\begin{array}{lllll}
k_1 = \mathbf{r} \cdot \mathbf{v} & k_2 = \mathbf{p} \cdot \mathbf{r} & k_3 = \mathbf{p} \cdot \mathbf{v} & k_4 = \mathbf{v} \cdot \mathbf{v} & s = r^\alpha \\
t1 = r^\alpha & t10 = \frac{k_1}{t3} & t109 = k_3 t17 & t12 = 1/2\alpha t2 t10 & t120 = t16 r \\
t131 = t56 t1 & t133 = t17 r^2 & t139 = p^2 & t14 = t2 t1 & t140 = \alpha t82 t139 \\
t142 = k_3 k_1 & t144 = t26^2 & t16 = \alpha - 1 & t162 = t24 t26 & t163 = t162 t62 \\
t166 = \frac{7}{24} & t168 = 2 t24 t26 & t17 = t3^2 & t178 = t24^2 & t179 = \alpha t178 \\
\frac{47}{12} t24 t26 & & & & \\
t18 = t17 r & t180 = \alpha - 4/3 & t182 = \alpha - 3/2 & t187 = t133^{-1} & t2 = t1^2 \\
t23 = \alpha^2 & t230 = t178 t16 & t231 = t180 t182 & t24 = k_1^2 & t25 = t23 t24 \\
t254 = \frac{52}{137} t26 t23 & = t26 = k_4 + k_2 & t263 = \frac{421}{274} t163 & t27 = 1/2 t26 t3 & t28 = 1/2 r \\
t283 = t56 t2 & t289 = k_1 t139 & t291 = k_3 t26 & t3 = r^2 & t33 = t17^{-1} \\
t334 = t23^2 & t355 = t178 k_1 & t359 = t231(\alpha - 8/5) & t364 = \frac{1}{t133 t3} & t39 = t24 t16 \\
t4 = t3 r & t56 = t2^2 & t62 = t23 \alpha & t66 = t24 k_1 & t69 = -1 + t26 r \\
t73 = \alpha - 6/7 & t78 = \alpha k_3 t17 & t82 = t17 t3 & t83 = t82^{-1} & t87 = \alpha t3 \\
\\
F_1 = 0 & G_1 = t1 & H_1 = 0 & T_1 = t1 & \\
F_2 = -1/2 \frac{t2}{t4} & G_2 = t12 & H_2 = 1/2 t2 & T_2 = t12 & \\
F_3 = -1/2 \frac{t14 k_1 t16}{t18} & & G_3 = 1/3(t25 + (t27 - t28 - t24)\alpha - t28)t14 t33 & & \\
H_3 = 1/2 t14 \alpha t10 & & T_3 = 1/3\alpha(t27 - t28 + t39)t14 t33 & & \\
\end{array}$$

$$\begin{aligned}
F_4 &= -\frac{11}{24} \left(4/11(\alpha - 3/4)t26 t3 + (2/11 - 4/11\alpha)r + t24 \left(\alpha - \frac{15}{11} \right) t16 \right) t56 t17^{-1} t4^{-1} \\
G_4 &= 1/4 t56 ((t62 - 7/3 t23 + 4/3\alpha)t66 + 7/6r(-1 + t69\alpha)t73 k_1 + 1/2 t78)t83 \\
H_4 &= 1/24 t56 (11 t25 + 4 t87 k_4 - 4\alpha r + 4 t87 k_2 - 8\alpha t24 - r)t33 \\
T_4 &= 1/4\alpha((t23 - 7/3\alpha + 4/3)t66 + 7/6r(t26 t73 r + 5/7 - \alpha)k_1 + 1/2 t109)t56 t83 \\
F_5 &= -\frac{5}{12} \left(\left(t62 - \frac{39}{10} t23 + 5\alpha - \frac{21}{10} \right) t66 + \frac{9}{10} t120 (t26 t16 r - \alpha + 2/3)k_1 + 3/10 t109(\alpha - 3/5) \right) t131 t133^{-1} r^{-1} \\
G_5 &= 1/5 \left(1/8 t140 + \frac{7}{24} \alpha \left(\left(\frac{33}{7} t142 + t144 \right) \alpha - \frac{24}{7} t142 - 6/7 t144 \right) t17 + \left(-\frac{7}{12} t26 t23 + (-1/12 k_2 + 1/24 k_4) \alpha + \right. \right. \\
&\quad \left. \left. 3/8 k_2 + 3/8 k_4 \right) t4 + \left(-1/3 + \frac{23}{12} t163 + t166 t23 + \left(\frac{5}{24} + t168 \right) \alpha \right) t3 - \frac{23}{12} t39 \left(t23 - 1/23 \alpha - \frac{45}{46} \right) r + \right. \\
&\quad \left. t179 t16 t180 t182 \right) t131 t187 \\
H_5 &= \frac{5}{12} t131 \left((t62 - 9/5 t23 + 4/5\alpha)t66 + \frac{9}{10} r(1/5 + t69 t23 + (1/3 - 2/3 t26 r)\alpha)k_1 + 3/10 t78 \right) t83 \\
T_5 &= 1/5\alpha \left(1/8 t82 t139 + \left(\left(\frac{11}{8} t142 + \frac{7}{24} t144 \right) \alpha - t142 - 1/4 t144 \right) t17 + \left(-\frac{7}{12} t26 \alpha + \frac{11}{24} k_4 + 1/3 k_2 \right) t4 + \right. \\
&\quad \left. \left(\frac{23}{12} t162 t23 + t166 \alpha - \frac{5}{24} + t168 \right) t3 - \frac{23}{12} t39 \left(\alpha - \frac{37}{46} \right) r + t230 t231 \right) t131 t187 \\
F_6 &= -\frac{137}{360} \left(\frac{9}{137} t139 (\alpha - 1/2) t82 + \left(\left(\frac{243}{274} t142 + \frac{26}{137} t144 \right) t23 + \left(-\frac{207}{137} t142 - \frac{48}{137} t144 \right) \alpha + \frac{90}{137} t142 + \right. \right. \\
&\quad \left. \left. \frac{45}{274} t144 \right) t17 + \left(t254 + \left(\frac{83}{137} k_4 + \frac{74}{137} k_2 \right) \alpha - \frac{57}{274} k_2 - \frac{33}{137} k_4 \right) t4 + \left(t263 + \left(\frac{26}{137} - \frac{722}{137} t24 t26 \right) t23 + \right. \right. \\
&\quad \left. \left. \left(-\frac{35}{137} + \frac{1653}{274} t24 t26 \right) \alpha + \frac{11}{137} - \frac{315}{137} t24 t26 \right) t3 - \frac{421}{274} t39 \left(t23 - \frac{857}{421} \alpha + \frac{420}{421} \right) r + t230 \left(t23 - \frac{416}{137} \alpha + \right. \right. \\
&\quad \left. \left. \frac{315}{137} \right) t182 \right) t283 t133^{-1} t4^{-1} \\
G_6 &= 1/6 t283 \left(5/8\alpha \left(\left(\frac{16}{25} t289 + t291 \right) \alpha - 2/5 t289 - 4/5 t291 \right) t82 - 5/8 k_3 \left(t23 - \frac{3}{25} \alpha - \frac{12}{25} \right) t18 + \frac{127}{120} \alpha k_1 \left(\left(\frac{303}{127} t142 + \right. \right. \right. \\
&\quad \left. \left. \left. t144 \right) t23 + \left(-\frac{534}{127} t142 - \frac{246}{127} t144 \right) \alpha + \frac{240}{127} t142 + \frac{120}{127} t144 \right) t17 - \frac{127}{60} \left(t26 t62 + \left(-\frac{119}{127} k_2 - \frac{143}{127} k_4 \right) t23 + \right. \right. \\
&\quad \left. \left. \left(-\frac{72}{127} k_4 - \frac{183}{254} k_2 \right) \alpha + \frac{90}{127} k_2 + \frac{90}{127} k_4 \right) k_1 t4 + \frac{163}{60} \left(\frac{75}{163} + t162 t334 + \left(\frac{127}{326} - \frac{557}{163} t24 t26 \right) t62 + \left(-\frac{20}{163} + \right. \right. \right. \\
&\quad \left. \left. \left. \frac{634}{163} t24 t26 \right) t23 + \left(-\frac{233}{326} - \frac{240}{163} t24 t26 \right) \alpha \right) k_1 t3 - \frac{163}{60} t66 \left(t62 - \frac{231}{163} t23 - \frac{130}{163} \alpha + \frac{210}{163} \right) t120 + \alpha t355 t16 t359 \right) t364 \\
\\
H_6 &= \frac{137}{360} \left(\frac{9}{137} t140 + \frac{26}{137} \alpha \left(\left(\frac{243}{52} t142 + t144 \right) \alpha - \frac{36}{13} t142 - \frac{9}{13} t144 \right) t17 + \left(t254 + \left(\frac{14}{137} k_2 + \frac{23}{137} k_4 \right) \alpha + \frac{9}{137} k_2 + \right. \right. \right. \\
&\quad \left. \left. \left. \frac{9}{137} k_4 \right) t4 + \left(-\frac{17}{274} + t263 + \left(\frac{26}{137} - \frac{347}{137} t24 t26 \right) t23 + \left(-\frac{5}{137} + \frac{144}{137} t24 t26 \right) \alpha \right) t3 - \frac{421}{274} t24 \left(\frac{47}{421} \alpha + t62 - \right. \right. \right. \\
&\quad \left. \left. \left. \frac{528}{421} t23 + \frac{90}{421} \right) r + t179 t16 \left(t23 - \frac{284}{137} \alpha + \frac{144}{137} \right) \right) t283 t187
\end{aligned}$$

$$\begin{aligned}
T_6 = & 1/6\alpha t^{283} \left((2/5 t^{139} (-5/8 + \alpha) k_1 + 5/8 (\alpha - 4/5) t^{26} k_3) t^{82} - 5/8 \left(\alpha - \frac{18}{25} \right) k_3 t^{18} + \frac{127}{120} k_1 \left(\frac{303}{127} \left(t^{23} - \frac{178}{101} \alpha + \frac{80}{101} \right) k_3 k_1 + t^{144} \left(\frac{120}{127} + t^{23} - \frac{246}{127} \alpha \right) \right) t^{17} - \frac{127}{60} \left(t^{26} t^{23} + \left(-\frac{403}{254} k_2 - \frac{451}{254} k_4 \right) \alpha + \frac{201}{254} k_4 + \frac{81}{127} k_2 \right) k_1 t^{14} + \right. \\
& \left. \frac{163}{60} k_1 \left(\left(t^{23} - \frac{394}{163} \alpha + \frac{240}{163} \right) t^{16} t^{162} + \frac{127}{326} t^{23} + \frac{41}{163} - \frac{205}{326} \alpha \right) t^3 - \frac{163}{60} t^{66} \left(t^{23} - \frac{687}{326} \alpha + \frac{355}{326} \right) t^{120} + t^{355} t^{16} t^{359} \right) t^{364}
\end{aligned}$$

Appendix D

List of Publications

D.1 Refereed Journal Publications

- Etienne Pellegrini**, Ryan P. Russell, On the Computation and Accuracy of Trajectory State-Transition Matrices, *Journal of Guidance, Controls and Dynamics*, Vol. 39 No. 11, pp. 2485-2499, November 2016, [doi:10.2514/1.G001920](https://doi.org/10.2514/1.G001920) Chapter 5
- Etienne Pellegrini**, Ryan P. Russell, F and G Series Solutions to the Kepler and Stark Problems with Sundman Transformations, *Celestial Mechanics and Dynamical Astronomy*, Vol. 118 No. 4, pp. 355-378, March 2014, [doi:10.1007/s10569-014-9538-7](https://doi.org/10.1007/s10569-014-9538-7) Chapter 6

D.2 Conference Proceedings

- Etienne Pellegrini** and Ryan P. Russell, A Multiple-Shooting Differential Dynamic Programming Algorithm, Paper AAS 17-453, *AAS/AIAA Space Flight Mechanics Meeting*, San Antonio, TX, February 2017 Chapter 3 & Chapter 4
- Etienne Pellegrini** and Ryan P. Russell, Quasi-Newton Differential Dynamic Programming for Robust Low-Thrust Optimization, *AIAA/AAS Astrodynamics Specialists Conference*, Minneapolis, MN, August 2012, [doi:10.2514/6.2012-4425](https://doi.org/10.2514/6.2012-4425) Chapter 3 & Chapter 4

- Etienne Pellegrini** and Ryan P. Russell, On the Accuracy of Trajectory State-Transition Matrices, Paper AAS 15-785, *AAS/AIAA Astrodynamics Specialists Conference*, Vail, CO, August 2015 [Chapter 5](#)
- Etienne Pellegrini** and Ryan P. Russell, F and G Taylor Series Solutions to the Circular Restricted Three-Body Problem, Paper AAS 14-237, *AAS/AIAA Space Flight Mechanics Meeting*, Santa Fe, NM, February 2014 [Chapter 6](#)
- Etienne Pellegrini** and Ryan P. Russell, F and G Taylor Series Solutions to the Stark Problem with Sundmann Transformations, Paper AAS 13-725, *AAS/AIAA Astrodynamics Specialists Conference*, Hilton Head, SC, August 2013 [Chapter 6](#)
- Haijun Shen, Vivek Vittaldev, Christopher Karlgaard, Ryan P. Russell, and **Etienne Pellegrini**, Parallelized Sigma Point and Particle Filters for Navigation Problems, Paper AAS 13-034, AAS G&C Conference, Breckenridge, CO, 2013 Others

Bibliography

- [1] J. Abadie and J. Carpentier. Generalization of the Wolfe Reduced-Gradient Method to the Case of Nonlinear Constraints. In Roger Fletcher, editor, *Optimization*, pages 37–49. Academic Press, London, 1969.
- [2] Rodney L. Anderson and Martin W. Lo. Flyby Design using Heteroclinic and Homoclinic Connections of Unstable Resonant Orbits. In *AAS/AIAA Spaceflight Mechanics Meeting*, 2011.
- [3] Michael Athans, Richard P. Wishner, and Anthony Bertolini. Suboptimal State Estimation for Continuous-Time Nonlinear Systems from Discrete Noisy Measurements. *IEEE Transactions on Automatic Control*, 13(5):504–514, 1968. doi: 10.1109/TAC.1968.1098986.
- [4] Eva Balsa Canto, Julio R. Banga, Antonio A. Alonso, and Vassilios S. Vassiliadis. Restricted Second Order Information for the Solution of Optimal Control Problems Using Control Vector Parameterization. *Journal of Process Control*, 12(2):243–255, 2002. doi: 10.1016/S0959-1524(01)00008-7. URL <http://www.sciencedirect.com/science/article/pii/S0959152401000087>.
- [5] Alex Barclay, Philip E. Gill, and J. Ben Rosen. SQP Methods and their Application to Numerical Optimal Control. In W. H. Schmidt,

- K. Heier, L. Bittner, and R. Bulirsch, editors, *Variational Calculus, Optimal Control and Applications*, volume 124 of *International Series of Numerical Mathematics*, pages 207–222. Birkhäuser Basel, Basel, 1998. ISBN 978-3-0348-9780-8. doi: 10.1007/978-3-0348-8802-8_21. URL http://link.springer.com/10.1007/978-3-0348-8802-8_21.
- [6] Roger R. Bate, Donald D. Mueller, and Jerry E. White. *Fundamentals of Astrodynamics*. Dover Publications, New-York, NY, 1971. ISBN 0486600610.
- [7] Richard H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics*. Education Series. American Institute of Aeronautics and Astronautics, Inc, 1999. ISBN 1563473429.
- [8] Joachim W. Baumgarte. Numerical Stabilization of the Differential Equations of Keplerian Motion. *Celestial Mechanics*, 5(4):490–501, 1972. doi: 10.1007/BF01464775.
- [9] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, 2006. ISBN 978-0-471-48600-8.
- [10] Richard Bellman. The Theory of Dynamic Programming. *Bulletin of the American Mathematical Society*, 60(6):503–516, nov 1954. ISSN 0002-9904. doi: 10.1090/S0002-9904-1954-09848-8. URL <http://www.ams.org/journal-getitem?pii=S0002-9904-1954-09848-8>.

- [11] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957. ISBN 0486428095.
- [12] Richard E. Bellman and Stuart E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1962. ISBN 0691625425. URL <http://www.rand.org/pubs/reports/R352.html?src=mobile>.
- [13] Mikhail A. Belyaev and Roman R. Rafikov. The Dynamics Of Dust Grains in the Outer Solar System. *The Astrophysical Journal*, 723(2):1718–1735, nov 2010. ISSN 0004-637X. doi: 10.1088/0004-637X/723/2/1718.
- [14] J. Bem and Barbara Szczodrowska-Kozar. High Order f and g Power Series for Orbit Determination. *Astronomy and Astrophysics Supplement*, 110:411–417, 1995.
- [15] Julio C. Benavides. *Trajectory Design using Approximate Analytic Solutions of the N-Body Problem*. Ph.d. thesis, The Pennsylvania State University, 2010.
- [16] Claus Bendtsen and Ole Stauning. FADBAD, a Flexible C++ Package for Automatic Differentiation. Technical report, Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1996.
- [17] D.A. Benson, G.T. Huntington, T.P. Thorvaldsen, and A.V. Rao. Direct Trajectory Optimization and Costate Estimation via an Orthogonal

- Collocation Method. *Journal of Guidance, Control, and Dynamics*, 29(6):1435–1440, 2006. ISSN 0731-5090. doi: 10.2514/1.20478.
- [18] P. Berkmann and Hans J. Pesch. Abort Landing in Windshear: Optimal Control Problem with Third-Order State Constraint and Varied Switching Structure. *Journal of optimization theory and applications*, 85(1):21–57, 1995. doi: 10.1007/BF02192298. URL <http://www.springerlink.com/index/pkm2q107k9t18757.pdf>.
- [19] Matthew M. Berry and Liam M. Healy. The Generalized Sundman Transformation for Propagation of High-Eccentricity Elliptical Orbits. *Advances in the Astronautical Sciences*, 112:127–146, jan 2002.
- [20] Matthew M. Berry and Liam M. Healy. Comparison of Accuracy Assessment Techniques for Numerical Integration. *Advances in the Astronautical Sciences*, 114:1003–1016, 2003. URL <http://hdl.handle.net/1903/3014>.
- [21] R. Bertrand and R. Epenoy. New smoothing techniques for solving bang-bang optimal control problems - Numerical results and statistical interpretation. *Optimal Control Applications and Methods*, 23(4):171–197, 2002. ISSN 01432087. doi: 10.1002/oca.709.
- [22] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, San Diego, 1982. ISBN 1886529043.
- [23] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific Publishers, 2nd ed edition, 1999. ISBN 978-1-886529-05-2.

- [24] John T. Betts. Optimal Interplanetary Orbit Transfers by Direct Transcription. *Journal of the Astronautical Sciences*, 42(3):247–268, 1994.
- [25] John T. Betts. Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998. doi: 10.2514/2.4231.
- [26] John T. Betts. Very Low-Thrust Trajectory Optimization Using a Direct SQP Method. *Journal of Computational and Applied Mathematics*, 120 (1-2):27–40, aug 2000. ISSN 03770427. doi: 10.1016/S0377-0427(00)00301-0. URL <http://linkinghub.elsevier.com/retrieve/pii/S0377042700003010>.
- [27] John T. Betts. *Practical Methods for Optimal Control using Nonlinear Programming*. Advances in Design and Control. SIAM, Philadelphia, 2001. doi: 10.11115/1.1483351.
- [28] John T. Betts. Optimal Lunar Swingby Trajectories. *The Journal of the Astronautical Sciences*, 55(3):349–371, sep 2007. ISSN 0021-9142. doi: 10.1007/BF03256529. URL <http://link.springer.com/10.1007/BF03256529>.
- [29] John T. Betts and William P. Huffman. Trajectory Optimization on a Parallel Processor. *Journal of Guidance, Control, and Dynamics*, 14(2):431–439, 1991. doi: 10.2514/3.20656.
- [30] John T. Betts and William P. Huffman. Application of Sparse Nonlinear Programming to Trajectory Optimization. *Journal of Guidance, Control,*

- and Dynamics*, 15(1):198–206, jan 1992. ISSN 0731-5090. doi: 10.2514/3.20819. URL <http://arc.aiaa.org/doi/10.2514/3.20819>.
- [31] John T. Betts and William P. Huffman. Sparse Optimal Control Software SOCS. Technical report, Boeing Information and Support Services, Seattle, WA, 1997.
 - [32] John T. Betts, T. Bauer, William P. Huffman, and K. Zondervan. Solving the Optimal Control Problem Using a Nonlinear Programming Technique. Part I: General Formulation. In *Astrodynamic Conference*, Seattle, WA, aug 1984. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.1984-2037. URL <http://dx.doi.org/10.2514/6.1984-2037>.
<http://arc.aiaa.org/doi/10.2514/6.1984-2037>.
 - [33] Lorenz T. Biegler. Solution of Dynamic Optimization Problems by Successive Quadratic Programming and Orthogonal Collocation. *Computers & Chemical Engineering*, 8(3-4):243–247, 1984. doi: 10.1016/0098-1354(84)87012-X.
 - [34] E. G. Birgin and J. M. Martinez. *Practical Augmented Lagrangian Methods for Constrained Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, may 2014. ISBN 978-1-61197-335-8. doi: 10.1137/1.9781611973365. URL <http://pubs.siam.org/doi/book/10.1137/1.9781611973365>.
 - [35] E. G. Birgin, R. A. Castillo, and J. M. Martinez. Numerical Comparison of Augmented Lagrangian Algorithms for Nonconvex Prob-

- lems. *Computational Optimization and Applications*, 31(1):31–55, may 2005. ISSN 0926-6003. doi: 10.1007/s10589-005-1066-7. URL <http://link.springer.com/10.1007/s10589-005-1066-7>.
- [36] George D. Birkhoff. The Restricted Problem of Three Bodies. *Rendiconti del Circolo Matematico di Palermo*, 39(1), 1915. doi: 10.1007/BF03015982.
- [37] Christian Bischof, Alan Carle, George Corliss, Andreas Griewank, and Paul D Hovland. ADIFOR: Generating Derivative Codes from Fortran Programs. *Scientific Programming*, 1(1):1–29, 1992. doi: 10.1155/1992/717832.
- [38] Sergio Bittanti and Patrizio Colaneri. Floquet Theory and Stability. In *Periodic Systems*, chapter 3, pages 81–108. Springer London, London, 2009. doi: 10.1007/978-1-84800-911-0_3.
- [39] Ake Björck and Victor Pereyra. Solution of Vandermonde Systems of Equations. *Mathematics of Computation*, 24(112):893–893, 1971. doi: 10.1090/S0025-5718-1970-0290541-1.
- [40] Jacek Błaszczyk, Andrzej Karbowski, and Krzysztof Malinowski. Object Library of Algorithms for Dynamic Optimization Problems: Benchmarking SQP and Nonlinear Interior Point Methods. *International Journal of Applied Mathematics and Computer Science*, 17(4):515–537, jan 2007. ISSN 1641-876X. doi: 10.2478/

v10006-007-0043-y. URL <http://www.degruyter.com/view/j/amcs.2007.17.issue-4/v10006-007-0043-y/v10006-007-0043-y.xml>.

- [41] Gilbert A. Bliss. *Lectures on the Calculus of Variations*. University of Chicago Press, Chicago, IL, 1946. ISBN 141818201X.
- [42] H. G. Bock and K. J. Plitt. A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems. In *IFAC 9th World Congress*, pages 242–247, Budapest, Hungary, 1984.
- [43] Hans G. Bock. Numerical Solution of Nonlinear Multipoint Boundary Value Problems with Applications to Optimal Control. *Z. Angew. Math. Mech.*, 58:407, 1978.
- [44] Victor R. Bond. A Recursive Formulation for Computing the Coefficients of the Time-Dependent f and g Series Solutions to the Two-Body Problem. *The Astronomical Journal*, 71(1):7–8, 1966. doi: 10.1086/109845.
- [45] Victor R. Bond. A Transformation of the Two-Body Problem. *Celestial Mechanics*, 35(1):1–7, 1985. doi: 10.1007/BF01229108.
- [46] George Boole. *A Treatise on the Calculus of Finite Differences*. Macmillan And Co, London, 3rd edition, 1880. doi: 10.1137/1003072.
- [47] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004. ISBN 9780511804441. doi: 10.1017/CBO9780511804441. URL <http://ebooks.cambridge.org/ref/id/CB09780511804441{%}5Cnhttp://>

<http://www.informaworld.com/openurl?genre=article&doi=10.1080/10556781003625177&magic=crossref>.

- [48] G. L. Brauer, D. E. Cornick, and R. Stevenson. Capabilities and Applications of the Program to Optimize and Simulate Trajectories. Technical report, National Aeronautics and Space Administration, 1977.
- [49] Roger A. Broucke. Periodic Orbits in the Restricted Three-Body Problem with Earth-Moon Masses. Technical report, Jet Propulsion Laboratory, Pasadena, California, 1968.
- [50] Roger A. Broucke. Stability of Periodic Orbits in the Elliptic, Restricted Three-Body Problem. *AIAA Journal*, 7(6):1003–1009, jun 1969. doi: 10.2514/3.5267.
- [51] Roger A. Broucke. Solution of the N-Body Problem with Recurrent Power Series. *Celestial Mechanics*, 4(1):110–115, 1971. doi: 10.1007/BF01230326.
- [52] C. G. Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of Computation*, 19(92):577, oct 1965. ISSN 00255718. doi: 10.2307/2003941. URL <http://www.jstor.org/stable/2003941?origin=crossref>.
- [53] Charles G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970. ISSN 0272-4960. doi: 10.1093/imamat/

- 6.1.76. URL <https://academic.oup.com/imamat/article-lookup/doi/10.1093/imamat/6.1.76>.
- [54] Victor A. Brumberg, S. V. Tarasevich, and N N Vasiliev. Specialized Celestial Mechanics Systems for Symbolic Manipulation. *Celestial Mechanics*, 45(1-3):149–162, 1988. doi: 10.1007/BF01228996.
- [55] Arthur E. Bryson and W. F. Denham. A Steepest-Ascent Method for Solving Optimum Programming Problems. *Journal of Applied Mechanics*, 29(2):247, 1962. ISSN 00218936. doi: 10.1115/1.3640537. URL <http://appliedmechanics.asmedigitalcollection.asme.org/article.aspx?articleid=1395107>.
- [56] Arthur E. Bryson and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere Pub, 1975. doi: 10.1109/TSMC.1979.4310229. URL [#0](http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Applied+Optimal+Control).
- [57] Roland Bulirsch, F. Montrone, and Hans J. Pesch. Abort Landing in the Presence of Windshear as a Minimax Optimal Control Problem. Part 1: Necessary Conditions. *Journal of Optimization Theory and Applications*, 70(1):1–23, 1991. doi: 10.1007/BF00940502.
- [58] Roland Bulirsch, F. Montrone, and Hans J. Pesch. Abort Landing in the Presence of Windshear as a Minimax Optimal Control Problem. Part 2: Multiple Shooting and Homotopy. *Journal of Optimization Theory*

- and Applications*, 70(2):223–254, 1991. doi: 10.1007/BF00940625. URL <http://link.springer.com/10.1007/BF00940625>.
- [59] C. Burrau. Über einige in Aussicht genommene Berechnung, betreffend einen Spezialfall des Dreikörperproblems. *Vierteljahrsschrift Astron. Ges.*, 41, 1906.
- [60] R. H. Byrd, R. A. Tapia, and Yin Zhang. An SQP Augmented Lagrangian BFGS Algorithm for Constrained Optimization. *SIAM Journal on Optimization*, 2(2):210–241, may 1992. ISSN 1052-6234. doi: 10.1137/0802012. URL <http://pubs.siam.org/doi/10.1137/0802012>.
- [61] Richard H. Byrd, Robert B. Schnabel, and Gerald A. Shultz. A Trust Region Algorithm for Nonlinearly Constrained Optimization. *SIAM Journal on Numerical Analysis*, 24(5):1152–1170, oct 1987. ISSN 0036-1429. doi: 10.1137/0724076. URL <http://pubs.siam.org/doi/10.1137/0724076>.
- [62] Richard H. Byrd, Jorge Nocedal, and Richard A. Waltz. Knitro: An Integrated Package for Nonlinear Optimization. In *Large-Scale Nonlinear Optimization*, volume 116 of *International Series in Operations Research & Management Science*, chapter 4, pages 35–59. Springer US, Boston, MA, 2006. ISBN 978-0-387-74502-2. doi: 10.1007/0-387-30065-1\4. URL <http://link.springer.com/10.1007/0-387-30065-1\4>.
- [63] Rainer Callies. Optimal Design of a Mission to Neptune. In Roland Bulirsch, A. Miele, J. Stoer, and K. H. Wells, editors, *Optimal Control*

- trol*, pages 341–349. Birkhäuser, Basel, Switzerland, 1993. doi: 10.1007/978-3-0348-7539-4_25.
- [64] Alan R. Campbell. *Numerical Analysis of Complex-Step Differentiation in Spacecraft Trajectory Optimization Problems*. Ms thesis, The University of Texas at Austin, 2011. URL <http://hdl.handle.net/2152/ETD-UT-2011-05-3440>.
- [65] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods in Fluid Dynamics*. Springer-Verlag, Heidelberg, Germany, 1988.
- [66] T. Carraro, M. Geiger, and R. Rannacher. Indirect Multiple Shooting For Nonlinear Parabolic Optimal Control Problems with Control Constraints. *SIAM Journal on Scientific Computing*, 36(2):A452–A481, 2014. doi: 10.1137/120895809.
- [67] Charles W. Carroll. *An Operations Research Approach to the Economic Optimization of a Kraft Pulping Process*. Ph.d. thesis, Institute of Paper Chemistry, Appleton, WI, 1959.
- [68] Charles W. Carroll. The Created Response Surface Technique for Optimizing Nonlinear, Restrained Systems. *Operations Research*, 9(2):169–184, apr 1961. ISSN 0030-364X. doi: 10.1287/opre.9.2.169. URL <http://pubsonline.informs.org/doi/abs/10.1287/opre.9.2.169>.
- [69] Augustin-Louis Cauchy. Methode Generale pour la Resolution des Sys-

- temes d'Equations Simultanées. Technical report, Academie des Sciences de Paris, Paris, 1847.
- [70] C. H. Chen, SC Chang, and I. Fong. An Effective Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. In *American Control Conference*, Pittsburgh, PA, 1989. IEEE. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4790479.
- [71] Thomas F. Coleman, Jianguo Liu, and Wei Yuan. A New Trust-Region Algorithm for Equality Constrained Optimization. *Journal of Computational Mathematics*, 21(2):177–199, 2002. doi: 10.1023/A:1013764800871.
- [72] A. R. Conn, N. Gould, A. Sartenaer, and Ph. L. Toint. Convergence Properties of an Augmented Lagrangian Algorithm for Optimization with a Combination of General Equality and Linear Constraints. *SIAM Journal on Optimization*, 6(3):674–703, aug 1996. ISSN 1052-6234. doi: 10.1137/S1052623493251463. URL <http://pubs.siam.org/doi/10.1137/S1052623493251463>.
- [73] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. Testing a Class of Methods for Solving Minimization Problems with Simple Bounds on the Variables. *Mathematics of Computation*, 50(182):399–430, 1988. doi: 10.2307/2008615. URL <http://www.jstor.org/stable/2008615>.
- [74] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. Global

- Convergence of a Class of Trust Region Algorithms for Optimization with Simple Bounds. *SIAM Journal on Numerical Analysis*, 25(2):433–460, 1988. ISSN 0036-1429. doi: 10.1137/0725029. URL <http://pubs.siam.org/doi/abs/10.1137/0725029>.
- [75] Andrew R. Conn, Nicholas I. M. Gould, and Philippe Toint. A Globally Convergent Augmented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, apr 1991. ISSN 0036-1429. doi: 10.1137/0728030. URL <http://pubs.siam.org/doi/abs/10.1137/0728030><http://pubs.siam.org/doi/10.1137/0728030>.
- [76] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. Convergence of Quasi-Newton Matrices Generated by the Symmetric Rank-One Update. *Mathematical programming*, 50:177–195, 1991. doi: 10.1007/BF01594934. URL <http://www.springerlink.com/index/w38h6v682kn11437.pdf>.
- [77] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Lancelot: A Fortran Package for Large-Scale Nonlinear Optimization*. Springer-Verlag, Berlin, 1992. doi: 10.1007/978-3-662-12211-2.
- [78] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. *Trust-Region Methods*, volume 1 of *SIAM Series on Optimization*. MPS - SIAM, Philadelphia, 2000. ISBN 0898714605. URL <http://books.google.com/books?hl=en&lr=>

- {&}id=5kNC4fqssYQC{&}oi=fnd{&}pg=PR15{&}dq=Trust-Region+
Methods{&}ots=j1KNQI1NJR{&}sig=ysnjX-TRkW8X5DpNbBgZTNKLrzA.
- [79] Bruce A. Conway. A Survey of Methods Available for the Numerical Optimization of Continuous Dynamic Systems. *Journal of Optimization Theory and Applications*, 152(2):271–306, feb 2012. ISSN 0022-3239. doi: 10.1007/s10957-011-9918-z. URL <http://link.springer.com/10.1007/s10957-011-9918-z>.
- [80] R Courant. Variational Methods for the Solution of Problems of Equilibrium and Vibrations. *Bulletin of the American Mathematical Society*, 49(1):1–24, jan 1943. ISSN 0002-9904. doi: 10.1090/S0002-9904-1943-07818-4. URL <http://projecteuclid.org/euclid.bams/1183504922http://www.ams.org/journal-getitem?pii=S0002-9904-1943-07818-4>.
- [81] Emiliano Cristiani and Pierre Martinon. Initialization of the Shooting Method via the Hamilton-Jacobi-Bellman Approach. *Journal of Optimization Theory and Applications*, 146(2):321–346, 2010. doi: 10.1007/s10957-010-9649-6.
- [82] Haskell B. Curry. The Method Of Steepest Descent. *Quarterly of Applied Mathematics*, 2(3):258–261, 1944.
- [83] William C. Davidon. Variable Metric Method for Minimization. *SIAM Journal on Optimization*, 1(1):1–17, feb 1991. ISSN 1052-6234. doi:

- 10.1137/0801001. URL <http://www.ii.uib.no/~lennart/drgrad/Davidon1991.pdf><http://pubs.siam.org/doi/10.1137/0801001>.
- [84] J. F. A. De O. Pantoja. Differential Dynamic Programming and Newton's Method. *International Journal of Control*, 47(5):1539–1553, may 1988. ISSN 0020-7179. doi: 10.1080/00207178808906114. URL <http://www.tandfonline.com/doi/abs/10.1080/00207178808906114>.
- [85] J.E. E. Dennis and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics, jan 1996. ISBN 978-0-89871-364-0. doi: 10.1137/1.9781611971200. URL <http://pubs.siam.org/doi/book/10.1137/1.9781611971200>.
- [86] John E. Dennis and Jorge J. More. Quasi-Newton Methods, Motivation and Theory. *SIAM review*, 19(1):46–89, 1977. doi: 10.1137/1019005.
- [87] Andre Deprit and J. F. Price. The Computation of Characteristic Exponents in the Planar Restricted Problem of Three Bodies. *The Astronomical Journal*, 70(10), 1965. doi: 10.1086/109823.
- [88] Andre Deprit and R. V. M. Zahar. Numerical Integration of an Orbit and Its Concomitant Variations by Recurrent Power Series. *Zeitschrift für angewandte Mathematik und Physik ZAMP*, 17(3):425–430, 1966. doi: 10.1007/BF01594535.
- [89] Andre Deprit, Antonio Elipe, and Sebastian Ferrer. Linearization:

- Laplace vs. Stiefel. *Celestial Mechanics and Dynamical Astronomy*, 58 (July 1992):151–201, 1994. doi: 10.1007/BF00695790.
- [90] P. Deuflhard, H. J. Pesch, and P. Rentrop. A Modified Continuation Method for the Numerical Solution of Nonlinear Two-Point Boundary Value Problems by Shooting Techniques. *Numerische Mathematik*, 26 (3):327–343, 1976. ISSN 0029599X. doi: 10.1007/BF01395950.
- [91] Peter Deuflhard. Recent Advances in Multiple Shooting Techniques. In Gladwell and Sayers, editors, *Computational Techniques for Ordinary Differential Equations*, pages 217–272. Academic Press, London, 1980. ISBN 0122857801.
- [92] Peter Deuflhard and G. Bader. Multiple Shooting Techniques Revisited. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, chapter 2, pages 74–94. Birkhäuser, Boston, 1983. doi: 10.1007/978-1-4684-7324-7__6.
- [93] Ernst D. Dickmanns and Klaus H. Well. Approximate Solution of Optimal Control Problems Using Third Order Hermite Polynomial Functions. In *Optimization Techniques IFIP Technical Conference*, pages 158–166, Novosibirsk, 1974. doi: 10.1007/3-540-07165-2__21. URL <http://www.springerlink.com/index/f1710p4122112315.pdf>
http://link.springer.com/10.1007/3-540-07165-2__21.
- [94] Moritz Diehl, Hans G. Bock, Holger Diedam, and Pierre-Brice Wieber. Fast Direct Multiple Shooting Algorithms for Optimal Robot Control.

Lecture Notes in Control and Information Sciences, 340:65–93, 2006. doi: 10.1007/978-3-540-36119-0\4.

- [95] Donald H. Ellison, Bruce A. Conway, and Jacob A. Englander. Numerical Computation of a Continuous-Thrust State Transition Matrix Incorporating Accurate Hardware and Ephemeris Models. In *AAS/AIAA Spaceflight Mechanics Conference*, Williamsburg, VA, 2015.
- [96] Gamal Elnagar, Mohammad A. Kazemi, and Mohsen Razzaghi. The Pseudospectral Legendre Method for Discretizing Optimal Control Problems. *IEEE Transactions on Automatic Control*, 40(10):1793–1796, 1995. ISSN 15582523. doi: 10.1109/9.467672.
- [97] P J Enright and B A Conway. Discrete Approximations to Optimal Trajectories Using Direct Transcription and Nonlinear Programming. *Journal of Guidance, Control, and Dynamics*, 15(4):994–1002, 1992. ISSN 0731-5090. doi: 10.2514/3.20934.
- [98] Fariba Fahroo and I. Ross. Trajectory optimization by indirect spectral collocation methods. In *AIAA/AAS Astrodynamics Specialist Conference*, pages 123–129, Reston, Virginia, aug 2000. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2000-4028. URL <http://dx.doi.org/10.2514/6.2000-4028><http://arc.aiaa.org/doi/10.2514/6.2000-4028>.
- [99] Fariba Fahroo and I. Michael Ross. Costate Estimation by a Legendre

- Pseudospectral Method. *Journal of Guidance, Control, and Dynamics*, 24(2):270–277, 2001. ISSN 0731-5090. doi: 10.2514/2.4709.
- [100] Fariba Fahroo and I. Michael 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. URL <http://arc.aiaa.org/doi/10.2514/2.4862>.
- [101] Terry Feagin and R. P. Mikkilineni. The Effect of Time Transformations on Local Truncation Errors. *Celestial Mechanics*, 13(4):491–493, 1976. doi: 10.1007/BF01229101.
- [102] Erwin Fehlberg. Numerical Integration of Differential Equations by Power Series Expansion. Technical report, NASA TN D-2356, 1964.
- [103] Sebastian Ferrer and Maria L. Sein-Echaluce. On the Szebehely-Bond Equation. General Sundman’s Transformation for the Perturbed Two-Body Problem. *Celestial Mechanics*, 32(4):333–347, 1984. doi: 10.1007/BF01229088.
- [104] Anthony V. Fiacco and Garth P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley & Sons, New York, 1968.
- [105] R. Fletcher. Function Minimization by Conjugate Gradients. *The Computer Journal*, 7(2):149–154, feb 1964. ISSN 0010-4620. doi: 10.1093/comjnl/7.2.149. URL <http://comjnl.oxfordjournals.org/content/7/2/149.short?rss=1&source=mfc%5Cnhttp://>

comjnl.oupjournals.org/cgi/doi/10.1093/comjnl/7.2.149
<https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/7.2.149>.

- [106] R. Fletcher. A New Approach to Variable Metric Algorithms. *The Computer Journal*, 13(3):317–322, mar 1970. ISSN 0010-4620. doi: 10.1093/comjnl/13.3.317. URL <https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/13.3.317>.
- [107] R. Fletcher. An Algorithm for Solving Linearly Constrained Optimization Problems. *Mathematical Programming*, 2(1):133–165, feb 1972. ISSN 0025-5610. doi: 10.1007/BF01584540. URL <http://link.springer.com/10.1007/BF01584540>.
- [108] R Fletcher and M J D Powell. A Rapidly Convergent Descent Method for Minimization. *The Computer Journal*, 6(2):163–168, 1963. ISSN 0010-4620. doi: 10.1093/comjnl/6.2.163. URL <http://comjnl.oxfordjournals.org/content/6/2/163.abstract>
<http://comjnl.oxfordjournals.org/cgi/doi/10.1093/comjnl/6.2.163>.
- [109] Roger Fletcher. *Practical Methods of Optimization*, 2nd ed. John Wiley & Sons, New-York, N.Y., 1987. ISBN 978-0-471-49463-8.
- [110] Marie Gaston Floquet. Sur les équations différentielles linéaires à co-

- efficients périodiques. *Annales Scientifiques de l'ENS*, 12:47–88, 1883.
ISSN 00015962.
- [111] Bernard H. Foing. Smart-1 Mission Overview from Launch, Lunar Orbit to Impact. *Lunar and Planetary Science*, XXXVIII, 2007. URL <http://www.lpi.usra.edu/meetings/lpsc2007/pdf/1915.pdf>.
- [112] Rodrigo Fontecilla, Trond Steihaug, and Richard A. Tapia. A Convergence Theory for a Class of Quasi-Newton Methods for Constrained Optimization. *SIAM Journal on Numerical Analysis*, 24(5):1133–1151, oct 1987. ISSN 0036-1429. doi: 10.1137/0724075. URL <http://pubs.siam.org/doi/abs/10.1137/0724075> <http://pubs.siam.org/doi/10.1137/0724075>.
- [113] John A. Ford and I. A. Moghrabi. Multi-Step Quasi-Newton Methods for Optimizaton. *Journal of Computational and Applied Mathematics*, 50(93):305–323, 1994. doi: 10.1016/0377-0427(94)90309-3.
- [114] John A. Ford and I. A. Moghrabi. Alternating Multi-Step Quasi-Newton Methods for Unconstrained Optimization. *Journal of Computational and Applied Mathematics*, 82(1-2):105–116, sep 1997. ISSN 03770427. doi: 10.1016/S0377-0427(97)00075-7. URL <http://linkinghub.elsevier.com/retrieve/pii/S0377042797000757>.
- [115] Bengt Fornberg. Numerical Differentiation of Analytic Functions. *ACM Trans. Math. Softw.*, 7(4):512–526, 1981. doi: 10.1145/355972.355979. URL <http://portal.acm.org/citation.cfm?id=355972.355979>.

- [116] Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, 51(184):699–699, 1988. doi: 10.1090/S0025-5718-1988-0935077-0.
- [117] Anders Forsgren, Philip E. Gill, and Margaret H. Wright. *Interior Methods for Nonlinear Optimization*, volume 44. 2002. ISBN 0036144502414. doi: 10.1137/S0036144502414942. URL <http://pubs.siam.org/doi/abs/10.1137/S0036144502414942>.
- [118] Ken Fox. Numerical Integration of the Equations of Motion of Celestial Mechanics. *Celestial Mechanics*, 33(2):127–142, 1984. doi: 10.1007/BF01234151.
- [119] Rüdiger Franke and Eckhard Arnold. Applying New Numerical Algorithms to the Solution of Discrete-Time Optimal Control Problems. In Miroslav Kárný and Kevin Warwick, editors, *Computer Intensive Methods in Control and Signal Processing*, pages 105–117. Birkhäuser Boston, Boston, MA, 1997. doi: 10.1007/978-1-4612-1996-5_.6. URL http://link.springer.com/10.1007/978-1-4612-1996-5_.6.
- [120] K. R. Frisch. The Logarithmic Potential Method of Convex Programming. Technical report, University Institute of Economics, Oslo, Norway, 1955.
- [121] Walter Gander. On Halley’s Iteration Method. *The American Mathematical Monthly*, 92(2):131–134, 1985. doi: 10.2307/2322644.

- [122] Divya Garg, Michael Patterson, William W. Hager, Anil V. Rao, David A. Benson, and Geoffrey T. Huntington. A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods. *Automatica*, 46(11):1843–1851, 2010. ISSN 00051098. doi: 10.1016/j.automatica.2010.06.048. URL <http://dx.doi.org/10.1016/j.automatica.2010.06.048>.
- [123] Stanley B. Gershwin and David H. Jacobson. A Discrete-Time Differential Dynamic Programming Algorithm with Application to Optimal Orbit Transfer. *AIAA Journal*, 8(9):1616–1626, 1970. doi: 10.2514/3.5955. URL <http://doi.aiaa.org/10.2514/3.5955>.
- [124] Philip E. Gill and Walter Murray. Newton-Type Methods for Unconstrained and Linearly Constrained Optimization. *Mathematical Programming*, 7(1):311–350, 1974. ISSN 00255610. doi: 10.1007/BF01585529.
- [125] Philip E. Gill and Walter Murray. *Numerical Methods for Constrained Optimization*. Academic Press, London, 1974.
- [126] Philip E. Gill and Elizabeth Wong. Sequential Quadratic Programming Methods. In *Mixed Integer Nonlinear Programming*, volume 154, pages 147–224. 2012. ISBN 978-1-4614-1926-6. doi: 10.1007/978-1-4614-1927-3_6. URL <http://link.springer.com/10.1007/978-1-4614-1927-3><http://link.springer.com/10.1007/978-1-4614-1927-3{ }6>.

- [127] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, 1981. ISBN 0122839528.
- [128] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. Some Theoretical Properties of an Augmented Lagrangian Merit Function. Technical report, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, CA, 1986.
- [129] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. Recent Developments in Constrained Optimization. *Journal of Computational and Applied Mathematics*, 22(2-3):257–270, jun 1988. ISSN 03770427. doi: 10.1016/0377-0427(88)90405-0.
- [130] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review*, 47(1):99–131, jan 2005. ISSN 0036-1445. doi: 10.1137/S0036144504446096. URL <http://pubs.siam.org/doi/10.1137/S0036144504446096>.
- [131] Philip E. Gill, Michael A. Saunders, and Elizabeth Wong. On the Performance of SQP Methods for Nonlinear Optimization. volume 4121, pages 95–123. 2015. doi: 10.1007/978-3-319-23699-5__5. URL http://link.springer.com/10.1007/978-3-319-23699-5__5.
- [132] Alexander Gofen. Interactive Environment for the Taylor Integration (in 3D Stereo). In *Proceedings of the 2005 International Conference on Scientific Computing (CSC 05)*. CSREA Press, 2005.

- [133] Donald Goldfarb. A Family of Variable-Metric Methods Derived by Variational Means. *Mathematics of Computation*, 24(109):23–23, jan 1970. ISSN 0025-5718. doi: 10.1090/S0025-5718-1970-0258249-6. URL <http://www.ams.org/jourcgi/jour-getitem?pii=S0025-5718-1970-0258249-6>.
- [134] Stephen M. Goldfeldt, Richard E. Quandt, Hale F. Trotter, Stephen M. Goldfeld, Richard E. Quandt, and Hale F. Trotter. Maximization by Quadratic Hill-Climbing. *Econometrica*, 34(3):541–551, jul 1966. ISSN 00129682. doi: 10.2307/1909768. URL <http://www.jstor.org/stable/1909768?origin=crossref>.
- [135] A. A. Goldstein. On Steepest Descent. *Journal of the Society for Industrial and Applied Mathematics Series A Control*, 3(1):147–151, jan 1965. ISSN 0887-4603. doi: 10.1137/0303013. URL <http://pubs.siam.org/doi/10.1137/0303013>.
- [136] Gerard Gomez, Wang S. Koon, Martin W. Lo, Jerrold E. Marsden, J. Masdemont, and Shane D. Ross. Invariant Manifolds, the Spatial Three-Body Problem and Space Mission Design. *Advances in Astronautical Sciences*, 109:3–22, 2001. URL <http://hdl.handle.net/2014/41546>.
- [137] Nicholas I. M. Gould, Jennifer a. Scott, and Yifan Hu. A Numerical Evaluation of Sparse Direct Solvers for the Solution of Large Sparse Symmetric Linear Systems of Equations.

- ACM Transactions on Mathematical Software*, 33(2):10–es, jun 2007. ISSN 00983500. doi: 10.1145/1236463.1236465. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.110.418&rep=rep1&type=pdf> <http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf> <http://portal.acm.org/citation.cfm?doid=1236463.1236465>.
- [138] Kathryn F. Graham and Anil V. Rao. Minimum-Time Trajectory Optimization of Multiple Revolution Low-Thrust Earth-Orbit Transfers. *Journal of Spacecraft and Rockets*, 52(3):711–727, may 2015. ISSN 0022-4650. doi: 10.2514/1.A33187. URL <http://www.scopus.com/inward/record.url?eid=2-s2.0-84930146891&partnerID=tZ0tx3y1http://arc.aiaa.org/doi/10.2514/1.A33187>.
- [139] John Greenstadt. On the Relative Efficiencies of Gradient Methods. *Mathematics of Computation*, 21(99):360, jul 1967. ISSN 00255718. doi: 10.2307/2003238. URL <http://www.jstor.org/stable/2003238?origin=crossref>.
- [140] A. Griewank and Ph L. Toint. Partitioned Variable Metric Updates for Large Structured Optimization Problems. *Numerische Mathematik*, 39(1):119–137, 1982. ISSN 0029599X. doi: 10.1007/BF01399316.
- [141] L. Grippo and S. Lucidi. Convergence Conditions, Line Search Algorithms and Trust-Region Implementations for the PolakRibi  re Conjugate Gradient Method. *Optimization Methods and Software*, 20(1):71–98, feb 2005. ISSN 1055-6788. doi: 10.1080/

1055678042000208570. URL <http://www.tandfonline.com/doi/abs/10.1080/1055678042000208570>.
- [142] K. G. Hadjifotinou and M. Gousidou-Koutita. Comparison of Numerical Methods for the Integration of Natural Satellite Systems. *Celestial Mechanics and Dynamical ...*, 70(2):99–113, 1998. doi: 10.1023/A:1026475908041. URL <http://link.springer.com/article/10.1023/A:1026475908041>.
- [143] D. W. Hahn and Forrester T. Johnson. Final Report on Chebychev Trajectory Optimization Program (CHEBYTOP 2). Technical report, The Boeing Co., Seattle, WA, 1971.
- [144] S. P. Han. A Globally Convergent Method for Nonlinear Programming. *Journal of Optimization Theory and Applications*, 22(3):297–309, 1977. doi: 10.1007/BF00932858. URL <http://www.springerlink.com/index/10.1007/BF00932858>.
- [145] Shih-Ping Han. Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems. *Mathematical Programming*, 11(1):263–282, dec 1976. ISSN 0025-5610. doi: 10.1007/BF01580395. URL <http://link.springer.com/10.1007/BF01580395>.
- [146] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4):338–342, jul 1987. ISSN 0731-5090. doi: 10.2514/3.20223. URL <http://arc.aiaa.org/doi/10.2514/3.20223>.

- [147] Mark Harris. Optimizing Parallel Reduction in CUDA. URL [http://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86{_\]website/projects/reduction/doc/reduction.pdf](http://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86{_]website/projects/reduction/doc/reduction.pdf).
- [148] Albert L. Herman and Bruce A. Conway. Direct Optimization Using Collocation Based on High-Order Gauss-Lobatto Quadrature Rules. *Journal of Guidance, Control, and Dynamics*, 19(3):592–599, may 1996. ISSN 0731-5090. doi: 10.2514/3.21662. URL <http://arc.aiaa.org/doi/10.2514/3.21662>.
- [149] Magnus R. Hestenes. *Calculus of Variations and Optimal Control Theory*. John Wiley and Sons, New York, 1966. ISBN 9780898740929.
- [150] Magnus R. Hestenes. Multiplier and Gradient Methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969. doi: 10.1007/BF00927673.
- [151] M.R. Hestenes and Eduard Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49(6):409, dec 1952. ISSN 0091-0635. doi: 10.6028/jres.049.044. URL [http://nvlpubs.nist.gov/nistpubs/jres/049/jresv49n6p409{_\]A1b.pdf](http://nvlpubs.nist.gov/nistpubs/jres/049/jresv49n6p409{_]A1b.pdf).
- [152] Nicholas J. Higham. The Accuracy of Floating Point Summation. *SIAM Journal on Scientific Computing*, 14(4):783–799, 1993. doi: 10.1137/0914050.

- [153] Tianyi Huang and K. A. Innanen. The Accuracy Check in Numerical Integration of Dynamical Systems. *The Astronomical Journal*, 88(6):870–876, 1983. doi: 10.1086/113374.
- [154] David G. Hull. Conversion of Optimal Control Problems into Parameter Optimization Problems. *Journal of Guidance, Control, and Dynamics*, 20(1):57–60, 1997. doi: 10.2514/2.4033.
- [155] David G. Hull. Optimal Control Theory for Applications, 2003.
- [156] David G. Hull and Walton E. Williamson. Numerical Derivatives for Parameter Optimization. *Journal of Guidance, Control, and Dynamics*, 2(2):158–160, 1979. doi: 10.2514/3.55854.
- [157] David H. Jacobson and David Q. Mayne. *Differential Dynamic Programming*. Elsevier Scientific, New-York, N.Y., 1970. ISBN 0444000704.
- [158] Forrester T. Johnson. Approximate Finite-Thrust Trajectory Optimization. *AIAA Journal*, 7(6):993–997, jun 1969. ISSN 0001-1452. doi: 10.2514/3.5265. URL <http://arc.aiaa.org/doi/10.2514/3.5265>.
- [159] Brandon A. Jones and Rodney L. Anderson. A Survey of Symplectic and Collocation Integration Methods for Orbit Propagation. In *AAS/AIAA Spaceflight Mechanics Conference*, pages 1–20, 2012.
- [160] Àngel Jorba and Maorong Zou. A Software Package for the Numerical Integration of ODEs by Means of High-Order Taylor Methods. *Experimental Mathematics*, 14(1):99–117, 2005. ISSN 1944-950X. doi: 10.1080/10586458.2005.10128904.

- [161] John L. Junkins, Manoranjan Majji, and James D. Turner. High Order Keplerian State Transition Tensors. In John L Crassidis, John L Junkins, Kathleen C Howell, Yaakov Oshman, and Julie K Thienel, editors, *Proceedings of the F. Landis Markley Astronautics Symposium*, Cambridge, Maryland, 2008. AAS.
- [162] N. Karmarkar. A New Polynomial-Time Algorithm for Linear Programming. *Combinatorica*, 4(4):373–395, dec 1984. ISSN 0209-9683. doi: 10.1007/BF02579150. URL <http://link.springer.com/10.1007/BF02579150>.
- [163] Jun’ichiro Kawaguchi, Hitoshi Kuninaka, Akira Fujiwara, and Tono Uesugi. MUSES-C, Its Launch and Early Orbit Operations. *Acta Astronautica*, 59(8-11):669–678, 2006. ISSN 00945765. doi: 10.1016/j.actaastro.2005.07.002.
- [164] Herbert B. Keller. *Numerical Methods for Two-Point Boundary Value Problems*. Blaisdell, Waltham, Mass, 1968. ISBN 0486669254.
- [165] Henry J. Kelley. Gradient Theory of Optimal Flight Paths. *ARS Journal*, 30(10):947–954, oct 1960. ISSN 1936-9972. doi: 10.2514/8.5282. URL <http://arc.aiaa.org/doi/10.2514/8.5282>.
- [166] H. Fayez Khalfan, Richard H. Byrd, and Robert B. Schnabel. A Theoretical and Experimental Study of the Symmetric Rank-One Update. *SIAM Journal on Optimization*, 3(1):1–24, 1993. doi: 10.1137/0803001. URL <http://locus.siam.org/doi/abs/10.1137/0803001>.

- [167] J. Kiefer. Sequential Minimax Search for a Maximum. *Proceedings of the American Mathematical Society*, 4(3):502–502, 1953. ISSN 0002-9939. doi: 10.1090/S0002-9939-1953-0055639-3. URL <http://www.ams.org/proc/1953-004-03/S0002-9939-1953-0055639-3/>.
- [168] Urs Kirchgraber. A Problem of Orbital Dynamics, Which is Separable in KS-Variables. *Celestial Mechanics*, 4(3):340–347, 1971. doi: 10.1007/BF01231396.
- [169] Christophe Koppel, Frederic Marchandise, Mathieu Prioul, Denis Estublier, and Franck Darnon. The Smart-1 Electric Propulsion Subsystem Around the Moon: In Flight Experience. In *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, number July, pages 1–10, Reston, Virginia, jul 2005. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-063-5. doi: 10.2514/6.2005-3671. URL <http://arc.aiaa.org/doi/abs/10.2514/6.2005-3671>.
- [170] Dieter Kraft. On Converting Optimal Control Problems into Nonlinear Programming Problems. In *Computational Mathematical Programming*, pages 261–280. Springer Berlin Heidelberg, Berlin, 1985. doi: 10.1007/978-3-642-82450-0__9.
- [171] H. Kreim, B. Kugelmann, H. J. Pesch, and M. H. Breitner. Minimizing the Maximum Heating of a Re-Entering Space Shuttle: An Optimal Control Problem with Multiple Control Constraints. *Optimal Control*

Applications and Methods, 17(1):45–69, jan 1996. doi: 10.1002/(SICI)1099-1514(199601/03)17:1<45::AID-OCA564>3.0.CO;2-X. URL [http://doi.wiley.com/10.1002/10.1002/\(SICI\)1099-1514\(199601/03\)17:1<45::AID-OCA564>3.0.CO;2-X](http://doi.wiley.com/10.1002/10.1002/(SICI)1099-1514(199601/03)17:1<45::AID-OCA564>3.0.CO;2-X).

- [172] H. W. Kuhn and A.W Tucker. Nonlinear Programming. *Proceedings of the Second Symposium on Mathematical Statistics and Probability*, (x):481–492, 1951. ISSN 01605682. doi: 10.1007/BF01582292. URL <http://projecteuclid.org/euclid.bsmsp/1200500249>.
- [173] Try Lam and Gregory J. Whiffen. Exploration of Distant Retrograde Orbits around Europa. In *AAS/AIAA Space Flight Mechanics Meeting*, Copper Mountain, CO, 2005.
- [174] Gregory Lantoine. *A Methodology for Robust Optimization of Low-Thrust Trajectories in Multi-Body Environments*. Ph.d. thesis, Georgia Institute of Technology, 2010.
- [175] Gregory Lantoine and Ryan Russell. A Hybrid Differential Dynamic Programming Algorithm for Robust Low-Thrust Optimization. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, volume 154, pages 382–417, Reston, Virigina, aug 2008. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-001-7. doi: 10.2514/6.2008-6615. URL <http://arc.aiaa.org/doi/10.2514/6.2008-6615>.

- [176] Gregory Lantoine and Ryan P. Russell. Complete Closed-Form Solutions of the Stark Problem. *Celestial Mechanics and Dynamical Astronomy*, 109(4):333–366, feb 2011. ISSN 0923-2958. doi: 10.1007/s10569-010-9331-1.
- [177] Gregory Lantoine and Ryan P. Russell. Near Ballistic Halo-to-Halo Transfers between Planetary Moons. *The Journal of the Astronautical Sciences*, 58(3):335–363, 2011. doi: 10.1007/BF03321174.
- [178] Gregory Lantoine and Ryan P. Russell. A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 1: Theory. *Journal of Optimization Theory and Applications*, 154(2):382–417, apr 2012. ISSN 0022-3239. doi: 10.1007/s10957-012-0039-0.
- [179] Gregory Lantoine and Ryan P. Russell. A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 2: Application. *Journal of Optimization Theory and Applications*, 154(2):418–442, apr 2012. doi: 10.1007/s10957-012-0038-1. URL <http://www.springerlink.com/index/10.1007/s10957-012-0038-1>.
- [180] Gregory Lantoine, Ryan P. Russell, and Stefano Campagnola. Optimization of Low-Energy Resonant Hopping Transfers Between Planetary Moons. *Acta Astronautica*, 68(7-8):1361–1378, 2011. ISSN 00945765. doi: 10.1016/j.actaastro.2010.09.021. URL <http://dx.doi.org/10.1016/j.actaastro.2010.09.021>.

- [181] Gregory Lantoine, Ryan P Russell, and Thierry Dargent. Using Multicomplex Variables for Automatic Computation of High-Order Derivatives. *ACM Transactions on Mathematical Software*, 38(3):16:1—16:21, apr 2012. doi: 10.1145/2168773.2168774. URL <http://doi.acm.org/10.1145/2168773.2168774>.
- [182] Derek F. Lawden. *Optimal Trajectories for Space Navigation*. Butterworth, London, 1963.
- [183] George Leitman. *The Calculus of Variations and Optimal Control*. Springer, New York, 1981. ISBN 0306407078.
- [184] Georges Lemaitre. Regularization of the Three Body Problem. *Vistas in Astronomy*, 1:207, 1955. doi: 10.1016/0083-6656(55)90028-3.
- [185] Kenneth Levenberg. A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly of Applied Mathematics*, 2(2): 164–168, 1944.
- [186] Tullio Levi-Civita. Sur la Regularisation du Probleme des Trois Corps. *Acta Mathematica*, 42:99–144, 1920. doi: 10.1007/BF02404404.
- [187] Li-zhi Liao. Optimal Control Approach For Large Scale Unconstrained Optimization Problems. Technical report, Department of Mathematics, Hong-Kong Baptist University, Hong-Kong, 1995.
- [188] Li Zhi Liao and Christine A. Shoemaker. Convergence in Unconstrained Discrete-Time Differential Dynamic Programming. *IEEE Transactions*

- on Automatic Control*, 36(6):692–706, 1991. ISSN 15582523. doi: 10.1109/9.86943.
- [189] Li-zhi Liao and Christine A. Shoemaker. Advantages of Differential Dynamic Programming Over Newton’s Method for Discrete-Time Optimal Control Problems. Technical report, Cornell University, 1992. URL <http://dspace.library.cornell.edu/handle/1813/5474>.
- [190] E. B. Lim, Y. Q. Liu, K. L. Teo, and J. B. Moore. Linear-Quadratic Optimal Control with Integral Quadratic Constraints. *Optimal Control Applications and Methods*, 20:79–92, 1999. doi: 10.1002/(SICI)1099-1514(199903/04)20:2<79::AID-OCA647>3.0.CO;2-3.
- [191] T. C. Lin and J. S. Arora. Differential Dynamic Programming Technique for Constrained Optimal Control. Part 1: Theoretical Development. *Computational Mechanics*, 9:27–40, 1991. doi: 10.1007/BF00369913.
- [192] T. C. Lin and J. S. Arora. Differential Dynamic Programming Technique For Constrained Optimal Control. Part 2: Structural Example. *Computational Mechanics*, 9(1):27–40, 1991. ISSN 0178-7675. doi: 10.1007/BF00369913. URL <http://link.springer.com/10.1007/BF00369913>.
- [193] F. A. Lootsma. A Survey of Methods for Solving Constrained Optimization Problems via Unconstrained Minimization. In F. A. Lootsma, editor, *Numerical Methods for Nonlinear Optimization*, pages 313 – 347. Academic Press, London, New-York, 1972.

- [194] David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*, volume 228 of *International Series in Operations Research & Management Science*. Springer International Publishing, 2016. ISBN 978-3-319-18841-6. doi: 10.1007/978-3-319-18842-3. URL <http://link.springer.com/10.1007/978-3-319-18842-3>.
- [195] Rein Luus. *Iterative Dynamic Programming*. Number 110 in Monographs and Surveys in Pure and Applied Mathematics. Chapman & Hall/CRC, 2000. ISBN 9781584881483. URL [#0.](http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:iterative+Dynamic+programming)
- [196] James N. Lyness. Numerical Algorithms Based on the Theory of Complex Variable. *ACM Proceedings*, pages 125–133, 1967. doi: 10.1145/800196.805983. URL <http://portal.acm.org/citation.cfm?doid=800196.805983>.
- [197] James N. Lyness and C. B. Moler. Numerical Differentiation of Analytic Functions. *SIAM Journal on Numerical Analysis*, 4(2):202–210, 1967. doi: 10.1145/355972.355979.
- [198] Manoranjan Majji, John L. Junkins, and James D. Turner. A High Order Method for Estimation of Dynamic Systems. *The Journal of the Astronautical Sciences*, 56(3):401–440, 2008. doi: 10.1007/BF03256560.
- [199] Donald W. Marquardt. An Algorithm For Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied*

- Mathematics*, 11(2):431–441, jun 1963. ISSN 0368-4245. doi: 10.1137/0111030. URL <http://pubs.siam.org/doi/10.1137/0111030>.
- [200] Joaquim R. R. A. Martins and John T. Hwang. Review and Unification of Methods for Computing Derivatives of Multidisciplinary Computational Models. *AIAA Journal*, 51(11):2582–2599, 2013. ISSN 0001-1452. doi: 10.2514/1.J052184.
- [201] Joaquim R. R. A. Martins, Peter Sturdza, and Juan J. Alonso. The Connection Between the Complex-Step Derivative Approximation and Algorithmic Differentiation. In *Proceedings of the 39th Aerospace Sciences Meeting and Exhibit*, Reno, NV, jan 2001. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2001-921. URL <http://arc.aiaa.org/doi/abs/10.2514/6.2001-921>.
- [202] Joaquim R. R. A. Martins, Peter Sturdza, and Juan J. Alonso. The Complex-Step Derivative Approximation. *ACM Transactions on Mathematical Software*, 29(3):245–262, sep 2003. doi: 10.1145/838250.838251. URL <http://dl.acm.org/citation.cfm?doid=838250.838251>.
- [203] David Mayne. A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems. *International Journal of Control*, 3(1):85–95, jan 1966. doi: 10.1080/00207176608921369. URL <http://www.tandfonline.com/doi/abs/10.1080/00207176608921369>.
- [204] T. Troy McConaghy, Theresa J. Debban, Anastassios E. Petropoulos,

- and James M. Longuski. Design and Optimization of Low-Thrust Trajectories with Gravity Assists. *Journal of Spacecraft and Rockets*, 40(3):380–387, may 2003. ISSN 0022-4650. doi: 10.2514/2.3973. URL <http://arc.aiaa.org/doi/10.2514/2.3973>.
- [205] A. Meghlaoui, R. Ouellet, and R.T. Bui. A General Structure of Specific Optimal Control. *Applied Mathematical Modelling*, 17(10):555–560, oct 1993. doi: 10.1016/0307-904X(93)90085-U. URL <http://linkinghub.elsevier.com/retrieve/pii/0307904X9390085U>.
- [206] Rainer Mehlhorn and Gottfried Sachs. A New Tool for Efficient Optimization by Automatic Differentiation and Program Transparency. *Optimization Methods and Software*, 4(3):225–242, jan 1994. ISSN 1055-6788. doi: 10.1080/10556789408805589. URL <http://www.tandfonline.com/doi/abs/10.1080/10556789408805589>.
- [207] R. H. Merson. Numerical Integration of the Differential Equations of Celestial Mechanics. Technical report, Royal Aircraft Establishment, Farnborough, Hants., England, 1973.
- [208] A. Miele. Recent Advances in Gradient Algorithms for Optimal Control Problems. *Journal of Optimization Theory and Applications*, 17(5-6):361–430, dec 1975. ISSN 0022-3239. doi: 10.1007/BF00932781. URL <http://link.springer.com/10.1007/BF00932781>.
- [209] A. Miele, P. E. Moseley, A. V. Levy, and G. M. Coggins. On the Method of Multipliers for Mathematical Programming Problems. *Journal of Op-*

- timization Theory and Applications*, 10(1):1–33, jul 1972. ISSN 0022-3239. doi: 10.1007/BF00934960. URL <http://link.springer.com/10.1007/BF00934960>.
- [210] Seppo Mikkola and David Merritt. Implementing Few-Body Algorithmic Regularization with Post-Newtonian Terms. *The Astronomical Journal*, 135(6):2398–2405, 2007. doi: 10.1088/0004-6256/135/6/2398. URL <http://arxiv.org/abs/0709.3367>.
- [211] Andrea Milani and Anna M. Nobili. Integration Error Over Very Long Time Spans. *Celestial Mechanics and Dynamical Astronomy*, 43:1–34, 1988. doi: 10.1007/BF01234550z.
- [212] Oliver Montenbruck. Numerical Integration of Orbital Motion using Taylor Series. In *AAS/AIAA Spaceflight Mechanics Conference*, Colorado Springs, CO, 1992. AAS. doi: 10.1007/BF00049361.
- [213] Oliver Montenbruck. Numerical Integration Methods for Orbital Motion. *Celestial Mechanics and Dynamical Astronomy*, 53(1):59 – 69, 1992. doi: 10.1007/BF00049361.
- [214] J. Morales, Jorge Nocedal, R. Waltz, G. Liu, and J. Goux. Assessing the Potential of Interior Methods for Nonlinear Optimization. *Large-Scale PDE-Constrained Optimization*, 30, 2003. URL <citeseer.ifi.unizh.ch/morales02assessing.html>.
- [215] Jorge J. More. Recent Developments in Algorithms and Software for Trust Region Methods. In A. Bachem, M. Grotschel, and B. Korte,

- editors, *Mathematical Programming The State of the Art*, pages 258–287. Springer Berlin Heidelberg, Berlin, Heidelberg, 1983. doi: 10.1007/978-3-642-68874-4_{11}. URL http://www.springerlink.com/index/10.1007/978-3-642-68874-4_{11}.
- [216] Jorge J. More and Danny C. Sorensen. Computing a Trust Region Step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983. doi: 10.1137/0904038.
 - [217] Daniel M. Murray and Sidney J. Yakowitz. The Application of Optimal Control Methodology to Nonlinear Programming Problems. *Mathematical Programming*, 21(1):331–347, dec 1981. ISSN 0025-5610. doi: 10.1007/BF01584253. URL <http://link.springer.com/10.1007/BF01584253>.
 - [218] Daniel M. Murray and Sidney J. Yakowitz. Differential Dynamic Programming and Newton’s Method for Discrete Optimal Control Problems. *Journal of Optimization Theory and Applications*, 43(3):395–414, jul 1984. ISSN 0022-3239. doi: 10.1007/BF00934463. URL <http://link.springer.com/10.1007/BF00934463>.
 - [219] Walter Murray. Ill-Conditioning in Barrier and Penalty Functions Arising in Constrained Nonlinear Programming. In *Princeton Mathematical Programming Symposium*, 1967.
 - [220] Walter Murray. Analytical Expressions for the Eigenvalues and Eigenvectors of the Hessian Matrices of Barrier and Penalty Functions. *Journal*

- of Optimization Theory and Applications*, 7(3):189–196, mar 1971. ISSN 0022-3239. doi: 10.1007/BF00932477. URL <http://link.springer.com/10.1007/BF00932477>.
- [221] Walter Murray. Sequential Quadratic Programming Methods for Large-Scale Problems. *Computational Optimization and Applications*, 7(1): 127–142, 1997. ISSN 09266003. doi: 10.1023/A:1008671829454. URL <http://link.springer.com/10.1023/A:1008671829454>.
- [222] Walter Murray. Some Aspects of Sequential Quadratic Programming Methods. In Lorenz T. Biegler, Thomas F. Coleman, Andrew R. Conn, and Fadil N. Santosa, editors, *Large-Scale Optimization with Applications*, volume 93 of *The IMA Volumes in Mathematics and its Applications*, chapter 2, pages 21–35. Springer New York, 1997. doi: 10.1007/978-1-4612-1960-6__2. URL http://link.springer.com/10.1007/978-1-4612-1960-6__2.
- [223] Bruce A. Murtagh and R. W. H. Sargent. A Constrained Minimization Method with Quadratic Convergence. In Robert Fletcher, editor, *Optimization*, pages 215–246. Academic Press, 1969.
- [224] Bruce A. Murtagh and Michael A. Saunders. A Projected Lagrangian Algorithm and its Implementation for Sparse Nonlinear Constraints. In A. G. Buckley and J.-L. Goffin, editors, *Algorithms for Constrained Minimization of Smooth Nonlinear Functions*, volume 16, chapter 5, pages 84–117. Springer, 1982. doi: 10.1007/BFb0120949.

- [225] Bruce A Murtagh and Michael A Saunders. MINOS 5.51 User's Guide. Technical report, Systems Optimization Laboratory, Stanford University, Stanford, CA, 2003.
- [226] Paul Nacozy. A Discussion of Time Transformations and Local Truncation Errors. *Celestial Mechanics*, 13(4):495–501, 1976. doi: 10.1007/BF01229102.
- [227] Paul Nacozy. The Intermediate Anomaly. *Celestial Mechanics*, 16(3):309–313, 1977. doi: 10.1007/BF01232657.
- [228] Fathi Namouni and Massimiliano Guzzo. The Accelerated Kepler Problem. *Celestial Mechanics and Dynamical Astronomy*, 99(1):31–44, aug 2007. ISSN 0923-2958. doi: 10.1007/s10569-007-9087-4.
- [229] Jorge Nocedal and Michael L. Overton. Projected Hessian Updating Algorithms for Nonlinearly Constrained Optimization. *SIAM Journal on Numerical Analysis*, 22(5):821–850, oct 1985. ISSN 0036-1429. doi: 10.1137/0722050. URL <http://www.jstor.org/stable/2157401> <http://epubs.siam.org/doi/abs/10.1137/0722050> <http://epubs.siam.org/doi/10.1137/0722050>.
- [230] Jorge Nocedal and Steven J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, New York, 2006. ISBN 978-0-387-30303-1. doi: 10.1007/978-0-387-40065-5. URL <http://link.springer.com/10.1007/978-0-387-40065-5>.

- [231] Jorge Nocedal, Andreas Wächter, and Richard A. Waltz. Adaptive Barrier Update Strategies for Nonlinear Interior Methods. *SIAM Journal on Optimization*, 19(4):1674–1693, jan 2009. ISSN 1052-6234. doi: 10.1137/060649513. URL <http://pubs.siam.org/doi/10.1137/060649513>.
- [232] Hans J. Oberle. Numerical Computation of Minimum-Fuel Space-Travel Problems by Multiple Shooting. Technical report, Technische Universität München, Munich, 1976.
- [233] Hans J. Oberle. On the Numerical Computation of Minimum-Fuel, Earth-Mars Transfer. *Journal of Optimization Theory and Applications*, 22(3):447–453, jul 1977. doi: 10.1007/BF00932866. URL <http://link.springer.com/10.1007/BF00932866>.
- [234] J. Oberle and W. Grimm. BNDSCO: A Program for the Numerical Solution of Optimal Control Problems. Technical report, Institute of Flight Systems Dynamics, German Aerospace Research Establishment DLR, Oberpfaffenhofen, Germany, 1990.
- [235] Cesar Ocampo and Juan Senent. The Design and Development of COPERNICUS: A Comprehensive Trajectory Design and Optimization System. In *57th International Astronautical Congress*, Reston, Virginia, oct 2006. American Institute of Aeronautics and Astronautics. ISBN 978-1-62410-042-0. doi: 10.2514/6.IAC-06-C1.4.04. URL <http://arc.aiaa.org/doi/abs/10.2514/6.IAC-06-C1.4.04>.
<http://arc.aiaa.org/doi/10.2514/6.IAC-06-C1.4.04>.

- [236] Cesar Ocampo, Juan S. Senent, and Jacob Williams. Theoretical Foundation of Copernicus: a Unified System for Trajectory Design and Optimization. In *4th International Conference on Astrodynamics Tools and Techniques*, Madrid, Spain, 2010.
- [237] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Society for Industrial and Applied Mathematics, jan 2000. ISBN 978-0-89871-461-6. doi: 10.1137/1.9780898719468. URL <http://pubs.siam.org/doi/book/10.1137/1.9780898719468>.
- [238] M. R. Osborne. On Shooting Methods for Boundary Value Problems. *Journal of Mathematical Analysis and Applications*, 27(2):417–433, 1969. doi: 10.1016/0022-247X(69)90059-6. URL <http://linkinghub.elsevier.com/retrieve/pii/0022247X69900596>.
- [239] A. M. Ostrowski. *Solutions of Equations in Euclidian and Banach Spaces*. Academic Press, New-York, N.Y., 3rd edition, 1973. ISBN 0125302606.
- [240] U. M. Garcia Palomares and O. L. Mangasarian. Superlinearly Convergent Quasi-Newton Algorithms for Nonlinearly Constrained Optimization Problems. *Mathematical Programming*, 11(1):1–13, dec 1976. ISSN 0025-5610. doi: 10.1007/BF01580366. URL <http://link.springer.com/10.1007/BF01580366>.
- [241] D. N. Papadakos. Generalized F and G Series and Convergence of the

- Power Series Solution to the N-Body Problem. *Celestial Mechanics*, 30(1):275–282, 1983. doi: 10.1007/BF01232193.
- [242] Ryan S. Park and Daniel J. Scheeres. Nonlinear Mapping of Gaussian Statistics: Theory and Applications to Spacecraft Trajectory Design. *Journal of Guidance, Control, and Dynamics*, 29(6):1367–1375, 2006. doi: 10.2514/1.20177.
- [243] Ryan S. Park and Daniel J. Scheeres. Nonlinear Semi-Analytic Methods for Trajectory Estimation. *Journal of Guidance, Control, and Dynamics*, 30(6):1668–1676, 2007. doi: 10.2514/1.29106.
- [244] Pavol Pástor. Influence of Fast Interstellar Gas Flow on the Dynamics of Dust Grains. *Celestial Mechanics and Dynamical Astronomy*, 112(1):23–45, oct 2011. ISSN 0923-2958. doi: 10.1007/s10569-011-9379-6.
- [245] Michael A. Patterson and Anil V. Rao. GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Non-linear Programming. *ACM Transactions on Mathematical Software*, 41(1):1–37, oct 2014. ISSN 00983500. doi: 10.1145/2558904. URL <http://dl.acm.org/citation.cfm?doid=2684421.2558904>.
- [246] J. D. Pearson. Variable Metric Methods of Minimisation. *The Computer Journal*, 12(2):171–178, feb 1969. ISSN 0010-4620. doi: 10.1093/comjnl/12.2.171. URL <https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/12.2.171>.

- [247] Etienne Pellegrini and Ryan P. Russell. On the Accuracy of Trajectory State-Transition Matrices. In *AAS/AIAA Astrodynamic Specialist Conference*, pages 3063–3082, Vail, CO, 2015. American Astronautical Society.
- [248] Etienne Pellegrini and Ryan P. Russell. On the Computation and Accuracy of Trajectory State Transition Matrices. *Journal of Guidance, Control, and Dynamics*, 39(11):2485–2499, nov 2016. doi: 10.2514/1.G001920. URL <http://arc.aiaa.org/doi/10.2514/1.G001920>.
- [249] Etienne Pellegrini and Ryan P Russell. A Multiple-Shooting Differential Dynamic Programming Algorithm. In *AAS/AIAA Space Flight Mechanics Meeting*, pages 759–778, San Antonio, TX, 2017.
- [250] Etienne Pellegrini and Ryan Paul Russell. Quasi-Newton Differential Dynamic Programming for Robust Low-Thrust Optimization. In *AIAA/AAS Astrodynamics Specialists Conference*, Minneapolis, MN, 2012. AIAA. doi: 10.2514/6.2012-4425. URL <http://arc.aiaa.org/doi/abs/10.2514/6.2012-4425>.
- [251] Etienne Pellegrini, Ryan P. Russell, and Vivek Vittaldev. F and G Taylor Series Solutions to the Stark Problem with Sundman Transformations. In *AAS/AIAA Astrodynamics Specialist Conference*, pages 369–388, Hilton Head, SC, 2013.
- [252] Etienne Pellegrini, Ryan P. Russell, and Vivek Vittaldev. F and G Taylor Series Solutions to the Stark and Kepler Problems with Sundman

- Transformations. *Celestial Mechanics and Dynamical Astronomy*, 118(4):355–378, 2014. doi: 10.1007/s10569-014-9538-7.
- [253] Hans J. Pesch. A Practical Guide to the Solution of Real-Life Optimal Control Problems. *Control and Cybernetics*, 23(1):7–60, 1994. ISSN 0324-8569.
- [254] E. Polak and G. Ribiere. Note sur la Convergence de Méthodes de Directions Conjuguées. *ESAIM: Mathematical Modelling and Numerical Analysis*, 3(16):35–43, 1969. ISSN 0764-583X. URL <https://eudml.org/doc/193115>.
- [255] E. Polak and A. L. Tits. A Globally Convergent, Implementable Multiplier Method with Automatic Penalty Limitation. *Applied Mathematics & Optimization*, 6(1):335–360, mar 1980. ISSN 0095-4616. doi: 10.1007/BF01442901. URL <http://link.springer.com/10.1007/BF01442901>.
- [256] Sergey M. Poleshchikov. One Integrable Case of the Perturbed Two-Body Problem. *Cosmic Research*, 42(4):398–407, jul 2004. ISSN 0010-9525. doi: 10.1023/B:COSM.0000039740.22909.ee.
- [257] Lev S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mischenko. *The Mathematical Theory of Optimal Processes*. John Wiley & Sons, New York, NY, 1962.
- [258] M. J D Powell. On Search Directions for Minimization Algorithms. *Mathematical Programming*, 4(1):193–201, dec 1973. ISSN 0025-5610.

- doi: 10.1007/BF01584660. URL <http://link.springer.com/10.1007/BF01584660>.
- [259] M. J D Powell and Y. Yuan. A Trust-Region Algorithm for Equality Constrained Optimization. *Mathematical Programming*, 49(1-3):189–211, 1990. ISSN 00255610. doi: 10.1007/BF01588787.
- [260] Michael J. D. Powell. A Method for Nonlinear Constraints in Minimization Problems. In R Fletcher, editor, *Optimization*, pages 283–298. Academic Press, 1969.
- [261] Michael J. D. Powell. A New Algorithm for Unconstrained Optimization. In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*, pages 31–65, Madison, WI, 1970. Academic Press, New York.
- [262] Michael J. D. Powell. Some Convergence Properties of the Conjugate Gradient Method. *Mathematical Programming*, 11(1):42–49, dec 1976. ISSN 0025-5610. doi: 10.1007/BF01580369.
URL <http://www.google.com/search?client=safari&rls=en-us&q=SOME+CONVERGENCE+PROPERTIES+OF+CONJUGATE+GRADIENT+METHOD&ie=UTF-8&oe=UTF-8%5Cnpapers://e21c739b-a592-49dd-9869-dfc7822cc78f/Paper/p6743http://link.springer.com/10.1007/BF01580369>.
- [263] Michael J. D. Powell. The Convergence of Variable Metric Methods for Nonlinearly Constrained Optimization Calculations. In O L Mangasar-

- ian, R R Meyer, and S M Robinson, editors, *Nonlinear Programming* 3, Nonlinear Programming, pages 27–63, Madison, WI, 1978. Academic Press, New York. doi: 10.1016/B978-0-12-468660-1.50007-4. URL [#7.](http://scholar.google.com/scholar?q=it+convergence&hl=en&btnG=Search&as_sdtp=on)
- [264] Michael J. D. Powell. A Fast Algorithm for Nonlinearly Constrained Optimization Calculations. In G Watson, editor, *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 144–157. Springer Berlin / Heidelberg, 1978. ISBN 978-3-540-08538-6. doi: 10.1007/BFb0067703. URL <http://dx.doi.org/10.1007/BFb0067703>.
- [265] Michael J. D. Powell. Algorithms for Nonlinear Constraints that use Lagrangian Functions. *Mathematical Programming*, 14(1):224–248, 1978. ISSN 00255610. doi: 10.1007/BF01588967.
- [266] Michael J. D. Powell. Variable Metric Methods for Constrained Optimization. In R Glowinski, J Lions, and Iria Laboria, editors, *Computing Methods in Applied Sciences and Engineering*, volume 704 of *Lecture Notes in Mathematics*, pages 62–72. Springer Berlin / Heidelberg, 1979. doi: 10.1007/BFb0063615. URL <http://onlinelibrary.wiley.com/doi/10.1002/cbdv.200490137/abstract> <http://www.springerlink.com/index/Y1K03G32756P0R6G.pdf> <http://dx.doi.org/10.1007/BFb0063615>.
- [267] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P

- Flannery. *Numerical Recipes in Fortran 77: the Art of Scientific Computing*. Cambridge University Press, 1992. ISBN 052143064X.
- [268] Eugene Rabe. Determination and Survey of Periodic Trojan Orbits in the Restricted Problem of Three Bodies. *The Astronomical Journal*, 66(9):500 – 513, 1961. doi: 10.1086/108451.
- [269] G.D Racca, A. Marini, L. Stagnaro, J van Dooren, L di Napoli, B.H Foing, R. Lumb, J. Volp, J. Brinkmann, R. Grünagel, D. Estublier, E. Tremolizzo, M. McKay, O. Camino, J. Schoemaekers, M. Hechler, M. Khan, P. Rathsman, G. Andersson, K. Anflo, S. Berge, P. Bodin, A. Edfors, A. Hussain, J. Kugelberg, N. Larsson, B. Ljung, L. Meijer, A. Mörtsell, T. Nordeback, S. Persson, and F. Sjöberg. SMART-1 Mission Description and Development Status. *Planetary and Space Science*, 50(14-15):1323–1337, dec 2002. ISSN 00320633. doi: 10.1016/S0032-0633(02)00123-X. URL <http://linkinghub.elsevier.com/retrieve/pii/S003206330200123X>.
- [270] J. E. Rader and David G. Hull. Computation of Optimal Aircraft Trajectories using Parameter Optimization Methods. *Journal of Aircraft*, 12(11):864–866, 1975. ISSN 0021-8669. doi: 10.2514/3.44497. URL <http://arc.aiaa.org/doi/10.2514/3.44497>.
- [271] A. Rakshit and S. Sen. Sequential Rank-One/Rank-Two Updates for Quasi-Newton Differential Dynamic Programming. *Optimal Control Applications and Methods*, 11:95–101, 1990. doi: 10.1002/oca.

4660110109. URL <http://onlinelibrary.wiley.com/doi/10.1002/oca.4660110109/abstract>.
- [272] Anil V. Rao. A Survey of Numerical Methods for Optimal Control. *Advances in the Astronautical Sciences*, 135(1):497–528, 2009. URL <http://vdol.mae.ufl.edu/ConferencePublications/trajectorySurveyAAS.pdf>.
- [273] Anil V. Rao. Trajectory Optimization: A Survey. In Harald Waschl, Ilya Kolmanovsky, Maarten Steinbuch, and Luigi del Re, editors, *Lecture Notes in Control and Information Sciences*, volume 455 of *Lecture Notes in Control and Information Sciences*, pages 3–21. Springer International Publishing, Cham, 2014. ISBN 978-3-319-05370-7. doi: 10.1007/978-3-319-05371-4__1. URL http://link.springer.com/10.1007/978-3-319-05371-4__1.
- [274] Anil V. Rao and Kenneth D. Mease. Dichotomic Basis Approach to solving Hyper-Sensitive Optimal Control Problems. *Automatica*, 35(4):633–642, apr 1999. doi: 10.1016/S0005-1098(98)00161-7.
- [275] Marc D. Rayman and Robert A. Mase. The Second Year of Dawn Mission Operations: Mars Gravity Assist and Onward to Vesta. *Acta Astronautica*, 67(3-4):483–488, 2010. ISSN 00945765. doi: 10.1016/j.actaastro.2010.03.010. URL <http://linkinghub.elsevier.com/retrieve/pii/S0094576510000998>.
- [276] Marc D. Rayman, Philip Varghese, David H. Lehman, and Leslie L.

- Livesay. Results From the Deep Space 1 Technology Validation Mission. *Acta Astronautica*, 47(2-9):475–487, jul 2000. ISSN 00945765. doi: 10.1016/s0094-5765(00)00087-4. URL <http://linkinghub.elsevier.com/retrieve/pii/S0094576500000874>.
- [277] Marc D. Rayman, Philip Varghese, David H. Lehman, and Leslie L. Livesay. Results from the Deep Space 1 Technology Validation Mission. *Acta Astronautica*, 47(2-9):475–487, jul 2000. ISSN 00945765. doi: 10.1016/S0094-5765(00)00087-4. URL <http://linkinghub.elsevier.com/retrieve/pii/S0094576500000874>.
- [278] Marc D. Rayman, Thomas C. Fraschetti, Carol A. Raymond, and Christopher T. Russell. Dawn: A Mission in Development for Exploration of Main Belt Asteroids Vesta and Ceres. *Acta Astronautica*, 58(11):605–616, 2006. doi: 10.1016/j.actaastro.2006.01.014. URL <http://linkinghub.elsevier.com/retrieve/pii/S0094576506000671>.
- [279] Ricardo L. Restrepo. *Automatic Algorithm for Accurate Numerical Gradient Calculation in General and Complex Spacecraft Trajectories*. PhD thesis, The University of Texas at Austin, 2013. URL <http://hdl.handle.net/2152/ETD-UT-2010-12-2624>.
- [280] J. P. Riehl, S. W. Paris, and W. K. Sjauw. Comparison of Implicit Integration Methods for Solving Aerospace Trajectory Optimization Problems. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, (August):1–26, 2006. doi: 10.2514/6.2006-6033. URL <http://www.aric.or.kr/treatise/journal/content.asp?idx=110007>.

- [281] S.M Roberts and J.S Shipman. Continuation in Shooting Methods for Two-Point Boundary Value Problems. *Journal of Mathematical Analysis and Applications*, 18(1):45–58, 1967. ISSN 0022247X. doi: 10.1016/0022-247X(67)90181-3. URL <http://www.sciencedirect.com/science/article/pii/0022247X67901813>.
- [282] CE Roberts, Jr. Comments on the Application of Power Series Solution to Problems in Celestial Mechanics. *Celestial Mechanics*, 12:397–407, 1975.
- [283] R. Tyrrell Rockafellar. The Multiplier Method of Hestenes and Powell Applied to Convex Programming. *Journal of Optimization Theory and Applications*, 12(6):555–562, 1973. doi: 10.1007/BF00934777.
- [284] R. Tyrrell Rockafellar. A Dual Approach to Solving Nonlinear Programming Problems by Unconstrained Optimization. *Mathematical Programming*, 5(1):354–373, dec 1973. doi: 10.1007/BF01580138. URL <http://link.springer.com/10.1007/BF01580138>.
- [285] R. Tyrrell Rockafellar. Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming. *Mathematics of Operations Research*, 1(2):97–116, 1976. URL <http://www.jstor.org/stable/3689277> <http://about.jstor.org/terms>.
- [286] A. J. Roenneke, C. Jansch, and A. Markl. Advanced Launch and Reentry Trajectory Optimization Software Documentation (ALTOS). Technical

report, European Space Science and Technology Center, Noordwijk, The Netherlands, 1995.

- [287] J. B. Rosen. The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints. *Journal of the Society for Industrial and Applied Mathematics*, 8(1):181–217, mar 1960. ISSN 0368-4245. doi: 10.1137/0108011. URL <http://pubs.siam.org/doi/10.1137/0108011>.
- [288] J B Rosen. The Gradient Projection Method for Nonlinear Programming. Part II. Nonlinear Constraints. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):514–532, dec 1961. ISSN 0368-4245. doi: 10.1137/0109044. URL <http://pubs.siam.org/doi/10.1137/0109044>.
- [289] I. Michael Ross and Fariba Fahroo. User’s Manual for DIDO 2001 α : A MATLAB Application for Solving Optimal Control Problems. Technical report, Department of Aeronautics and Astronautics, Naval Postgraduate School, Monterey, CA, 2001.
- [290] I Michael Ross and Fariba Fahroo. Legendre Pseudospectral Approximations of Optimal Control Problems. *New Trends in Nonlinear Dynamics and Control and their Applications*, 295(January):327–342, 2003. doi: 10.1007/978-3-540-45056-6_21. URL <http://www.springerlink.com/index/mmcgta9q1xmamvwq.pdf>.

- [291] I. Michael Ross and Mark Karpenko. A Review of Pseudospectral Optimal Control: From Theory to Flight. *Annual Reviews in Control*, 36(2):182–197, dec 2012. ISSN 13675788. doi: 10.1016/j.arcontrol.2012.09.002. URL <http://dx.doi.org/10.1016/j.arcontrol.2012.09.002><http://linkinghub.elsevier.com/retrieve/pii/S1367578812000375>.
- [292] Daniel Rufer. Trajectory Optimization by Making Use of the Closed Solution of Constant Thrust-Acceleration Motion. *Celestial Mechanics*, 14(1):91–103, 1976.
- [293] R. D. Russell and L. F. Shampine. A Collocation Method for Boundary Value Problems. *Numerische Mathematik*, 19(1):1–28, feb 1972. ISSN 0029-599X. doi: 10.1007/BF01395926. URL <http://link.springer.com/article/10.1007/BF01395926><http://link.springer.com/10.1007/BF01395926>.
- [294] Ryan P. Russell. Primer Vector Theory Applied to Global Low-Thrust Trade Studies. *Journal of Guidance, Control, and Dynamics*, 30(2):460–472, mar 2007. ISSN 0731-5090. doi: 10.2514/1.22984. URL <http://arc.aiaa.org/doi/10.2514/1.22984>.
- [295] David J. W. Ruxton. Differential Dynamic Programming Applied to Continuous Optimal Control Problems with State Variable Inequality Constraints. *Dynamics and Control*, 3(2):175–185, apr 1993. ISSN 0925-4668. doi: 10.1007/BF01968530. URL <http://link.springer.com/10.1007/BF01968530>.

- [296] R. W. H. Sargent. Reduced-Gradient and Projection Methods for Non-linear Programming. In Philip E. Gill and Walter Murray, editors, *Numerical Methods for Nonlinear Optimization*, chapter 5, pages 149–174. Academic Press, London, New-York, 1974.
- [297] R. W. H. Sargent. The Development of the SQP Algorithm for Non-linear Programming. In Lorenz T. Biegler, Thomas F. Coleman, Andrew R. Conn, and Fadil F. Santosa, editors, *Large-Scale Optimization with Applications*, volume 93 of *The IMA Volumes in Mathematics and its Applications*, chapter 1, pages 1–19. Springer New York, 1997. doi: 10.1007/978-1-4612-1960-6\1. URL <http://link.springer.com/10.1007/978-1-4612-1960-6\1>.
- [298] R. W. H. Sargent and G. R. Sullivan. The Development of an Efficient Optimal Control Package. In *Optimization Techniques*, volume 32, pages 158–168. Springer-Verlag, Berlin/Heidelberg, 1971. doi: 10.1007/BFb0006520. URL <http://www.springerlink.com/index/10.1007/BFb0006520>.
- [299] Klaus Schittkowski. The Nonlinear Programming Method of Wilson, Han, and Powell with an Augmented Lagrangian Type Line Search Function. *Numerische Mathematik*, 38(1):83–114, feb 1982. ISSN 0029-599X. doi: 10.1007/BF01395810. URL <http://link.springer.com/10.1007/BF01395810>.
- [300] P Sconzo, AR LeSchack, and R Tobey. Symbolic Computation of f and

- g Series by Computer. *The Astronomical Journal*, 70(4):269 – 270, 1965.
doi: 10.1086/109718.
- [301] James R Scott and Michael C Martini. High-Speed Solution of Spacecraft Trajectory Problems Using Taylor Series Integration. *Journal of Spacecraft and Rockets*, 47(1):199–202, 2010.
- [302] S. Sen and Sidney J. Yakowitz. A Quasi-Newton Differential Dynamic Programming Algorithm for Discrete-Time Optimal Control. *Automatica*, 23(6):749–752, 1987. doi: 10.1016/0005-1098(87)90031-8. URL <http://www.sciencedirect.com/science/article/pii/0005109887900318>.
- [303] Lawrence F. Shampine. Accurate Numerical Derivatives in MATLAB. *ACM Transactions on Mathematical Software*, 33(4):26–es, 2007. doi: 10.1145/1268776.1268781.
- [304] D. F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–647, sep 1970. ISSN 0025-5718. doi: 10.1090/S0025-5718-1970-0274029-X. URL <http://www.ams.org/jourcgi/jour-getitem?pii=S0025-5718-1970-0274029-X>.
- [305] M.A. Sharifi and M.R. Seif. Dynamic orbit propagation in a gravitational field of an inhomogeneous attractive body using the Lagrange coefficients. *Advances in Space Research*, 48(5):904–913, sep 2011. ISSN 02731177. doi: 10.1016/j.asr.2011.04.021.

- [306] Shi-Chung Chang, Chun-Hung Chen, I-Kong Fong, and P.B. Luh. Hydroelectric Generation Scheduling with an Effective Differential Dynamic Programming Algorithm. *IEEE Transactions on Power Systems*, 5(3):737–743, 1990. ISSN 08858950. doi: 10.1109/59.65900. URL <http://ieeexplore.ieee.org/document/65900/>.
- [307] Gerald A. Shultz, Robert B. Schnabel, and Richard H. Byrd. A Family of Trust-Region-Based Algorithms for Unconstrained Minimization with Strong Global Convergence Properties. *SIAM Journal on Numerical Analysis*, 22(1):47–67, feb 1985. ISSN 0036-1429. doi: 10.1137/0722003. URL <http://pubs.siam.org/doi/10.1137/0722003>.
- [308] Jon A Sims and Steve N Flanagan. Preliminary Design of Low-Thrust Interplanetary Missions. In *AAS/AIAA Astrodynamics Specialist Conference*, Girdwood, AK, USA, 1999.
- [309] Jon A. Sims, Paul A. Finlayson, Edward A. Rinderle, Matthew A. Vavrina, and Theresa D. Kowalkowski. Implementation of a Low-Thrust Trajectory Optimization Algorithm for Preliminary Design. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, (August):1–10, 2006. doi: 10.2514/6.2006-6746. URL <http://arc.aiaa.org/doi/abs/10.2514/6.2006-6746>.
- [310] T T Soong and N A Paul. A Second- and Higher Order Perturbation Analysis of Two-Body Trajectories. *AIAA Journal*, 9(4):589–593, 1971.
- [311] William Squire and George Trapp. Using Complex Variables to Estimate

- Derivatives of Real Functions. *SIAM Review*, 40(1):110–112, jan 1998.
doi: 10.1137/S003614459631241X. URL <http://pubs.siam.org/doi/abs/10.1137/S003614459631241X>.
- [312] Johannes Stark. Beobachtungen über den Effekt des elektrischen Feldes auf Spektrallinien. *Annalen der Physik*, 348(7):965–982, 1914.
- [313] J F Steffensen. On the Restricted Problem of Three Bodies. *Kongelige Danske Videnskabernes Selskab Mat.-Fys. Medd.*, 30(18), 1956.
- [314] E Stiefel. Remarks on Numerical Integration of Keplerian Orbits. *Celestial Mechanics*, 2(5):274–281, 1970.
- [315] Josef Stoer and Roland Bulirsch. *Introduction to Numerical Analysis*. Springer-Verlag, New York, 3 edition, 2002. ISBN 978-0-387-95452-3.
doi: 10.1007/978-0-387-21738-3.
- [316] Karl F. Sundman. Mémoire sur le problème des trois corps. *Acta Mathematica*, 36(1):105–179, dec 1912. ISSN 0001-5962. doi: 10.1007/BF02422379.
- [317] V Szebehely and V Bond. Transformations of the Perturbed Two-Body Problem to Unperturbed Harmonic Oscillators. *Celestial Mechanics*, 30(1):59–69, 1983.
- [318] V Szebehely and DA Pierce. Advantages of Regularization in Space Dynamics. *AIAA Journal*, 5(8):1520–1522, 1967.

- [319] Victor G. Szebehely. *Theory of Orbits, The Circular Restricted Three-Body Problem*. Academic Press, New York, NY, 1967.
- [320] Victor G. Szebehely and Frederick Peters. Complete Solution of a General Problem of Three Bodies. *The Astronomical Journal*, 72(7):876 – 883, 1967. doi: 10.1086/110355.
- [321] Richard A. Tapia. Diagonalized Multiplier methods and quasi-Newton Methods for Constrained Optimization. *Journal of Optimization Theory and Applications*, 22(2):135–194, 1977. doi: 10.1007/BF00933161.
- [322] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, oct 2012. doi: 10.1109/IROS.2012.6386025. URL <http://ieeexplore.ieee.org/abstract/document/6386025/>http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6386025.
- [323] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-Limited Differential Dynamic Programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, may 2014. doi: 10.1109/ICRA.2014.6907001. URL <http://ieeexplore.ieee.org/document/6907001/>.
- [324] T. N. Thiele. Recherche numerique concernant des solutions periodiques

d'un cas special du probleme des trois corps. *Astron. Nachr.*, 138(1), 1896.

- [325] F. Topputo and C. Zhang. Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications. *Abstract and Applied Analysis*, 2014:1–15, 2014. ISSN 1085-3375. doi: 10.1155/2014/851720.
URL <http://www.hindawi.com/journals/aaa/2014/851720/>.
- [326] James D. Turner. Automated Generation of High-Order Partial Derivative Models. *AIAA Journal*, 41(8):1590–1598, 2003. doi: 10.2514/2.2112.
- [327] James D. Turner, Manoranjan Majji, and John L. Junkins. High-Order State and Parameter Transition Tensor Calculations. In *AIAA/AAS Astrodynamics Specialist Conference*, Honolulu, Hawaii, 2008. American Institute of Aeronautics and Astronautics. ISBN 9781563479458. doi: 10.2514/6.2008-6453. URL <http://arc.aiaa.org/doi/pdf/10.2514/6.2008-6453>.
- [328] K. T. Uesugi. Space Engineering Spacecraft (MUSES) Program in ISAS Featuring its Latest Mission "Hayabusa". *RAST 2003 - Proceedings of International Conference on Recent Advances in Space Technologies*, pages 464–471, 2003. doi: 10.1109/RAST.2003.1303961.
- [329] Hodei Urrutxua, Manuel Sanjurjo-Rivo, and J Peláez. DROMO Propagator Revisited. *Celestial Mechanics and Dynamical Astronomy*, 124(1):1–31, 2015. doi: 10.1007/s10569-015-9647-y.

- [330] Avi Vardi. A Trust Region Algorithm for Equality Constrained Minimization: Convergence Properties and Implementation. *SIAM Journal on Numerical Analysis*, 22(3):575–591, jun 1985. ISSN 0036-1429. doi: 10.1137/0722035. URL <http://pubs.siam.org/doi/10.1137/0722035>.
- [331] Avi Vardi. A Trust Region Algorithm for Equality Constrained Minimization: Convergence Properties and Implementation. *SIAM Journal on Numerical Analysis*, 22(3):575–591, jun 1985. ISSN 0036-1429. doi: 10.1137/0722035. URL <http://pubs.siam.org/doi/10.1137/0722035>.
- [332] C E Velez. Notions of Analytic vs. Numerical Stability as applied to the Numerical Calculation of Orbits. *Celestial Mechanics*, 10(4):405–422, 1974. doi: 10.1007/BF01229118.
- [333] W. G. Vlases, S. W. Paris, R. M. Lajoie, M. J. Martens, and C. R. Har- graves. Optimal Trajectories by Implicit Simulation. Technical report, Boeing Aerospace and Electronics, Wright- Patterson Air Force Base, OH, 1990.
- [334] Oskar Von Stryk. Numerical Solution of Optimal Control Problems by Direct Collocation. In Roland Bulirsch, A. Miele, Josef Stoer, and K. H. Well, editors, *Optimal Control*, volume 111 of *International Series of Numerical Mathematics*, pages 129 – 143. Birkhäuser Basel, 1993. doi: 10.1007/978-3-0348-7539-4__10. URL http://link.springer.com/chapter/10.1007/978-3-0348-7539-4__10.

- [335] Oskar Von Stryk and Roland Bulirsch. Direct and Indirect Methods for Trajectory Optimization. *Annals of Operations Research*, 37:357–373, 1992. doi: 10.1007/BF02071065.
- [336] Andreas Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon, 2002.
- [337] Andreas Wächter and Lorenz T. Biegler. Line Search Filter Methods for Nonlinear Programming: Motivation and Global Convergence. *SIAM Journal on Optimization*, 16(1):1–31, jan 2005. ISSN 1052-6234. doi: 10.1137/S1052623403426556. URL <http://pubs.siam.org/doi/10.1137/S1052623403426556>.
- [338] Andreas Wächter and Lorenz T. Biegler. Line Search Filter Methods for Nonlinear Programming: Local Convergence. *SIAM Journal on Optimization*, 16(1):32–48, jan 2005. ISSN 1052-6234. doi: 10.1137/S1052623403426544. URL <http://pubs.siam.org/doi/10.1137/S1052623403426544>.
- [339] Andrea Walther. Getting Started with ADOL-C. In U Naumann and O Schenk, editors, *Combinatorial Scientific Computing*, chapter 7, pages 181–202. Chapman-Hall CRC Computational Science, Boca Raton, FL, 2012.
- [340] Gregory J. Whiffen. Static/Dynamic Control for Optimizing a Useful Objective, 2002.

- [341] Gregory J. Whiffen. Optimal Low-Thrust Orbital Transfers Around a Rotating Non-Spherical Body. *Advances in the Astronautical Sciences*, 119:2571–2586, 2005. ISSN 00653438.
- [342] Gregory J Whiffen. Mystic: Implementation of the Static Dynamic Optimal Control Algorithm for High-Fidelity, Low-Thrust Trajectory Design. In *AIAA/AAS Astrodynamics Specialists Conference*, 2006. doi: 10.2514/6.2006-6741.
- [343] Gregory J. Whiffen and Try Lam. The Jupiter Icy Moons Orbiter Reference Trajectory. *Advances in the Astronautical Sciences*, 124 II:1415–1436, 2006.
- [344] Gregory J Whiffen and Jon A Sims. Application of a Novel Optimal Control Algorithm to Low-Thrust Trajectory Optimization. In *Proceeding of the 11th Annual AAS/AIAA Space Flight Mechanics Meeting*, pages 1525–1540, 2001. URL <http://trs-new.jpl.nasa.gov/dspace/handle/2014/16157>.
- [345] W. W. Willman. On Second-Order Filter Performance. *Automatica*, 21(3):333–336, 1985. doi: 10.1016/0005-1098(85)90068-8.
- [346] R. B. Wilson. *A Simplicial Method for Convex Programming*. PhD thesis, Harvard University, 1963.
- [347] Philip Wolfe. Methods of Nonlinear Programming. In R. L. Graves and Philip Wolfe, editors, *Recent Advances in Mathematical Programming*. McGraw-Hill, New York, NY, 1963.

- [348] Philip Wolfe. Convergence Conditions for Ascent Methods. *SIAM Review*, 11(2):226–235, 1969. ISSN 0036-1445. doi: 10.1137/1013035. URL <http://pubs.siam.org/doi/abs/10.1137/1013035>.
- [349] Sidney Yakowitz. Dynamic Programming Applications in Water Resources. *Water Resources Research*, 18(4):673–696, aug 1982. ISSN 00431397. doi: 10.1029/WR018i004p00673. URL <http://doi.wiley.com/10.1029/WR018i004p00673>.
- [350] Sidney Yakowitz and Brian Rutherford. Computational Aspects of Discrete-Time Optimal Control. *Applied Mathematics and Computation*, 15(1):29–45, jul 1984. ISSN 00963003. doi: 10.1016/0096-3003(84)90051-1. URL <http://linkinghub.elsevier.com/retrieve/pii/0096300384900511>.
- [351] Sidney J. Yakowitz. Theoretical and Computational Advances in Differential Dynamic Programming. *Control and Cybernetics*, 17(2-3):173–189, 1988.
- [352] Sidney J. Yakowitz. *Algorithms and Computational Techniques in Differential Dynamic Programming*, volume 31. 1989.
- [353] Chit Hong Yam, Dario Izzo, and Francesco Biscani. Towards a High Fidelity Direct Transcription Method for Optimisation of Low-Thrust Trajectories. In *4th International Conference on Astrodynamics Tools and Techniques*, pages 1–7, 2010.

- [354] Hiroshi Yamashita. A Globally Convergent Constrained Quasi-Newton Method with an Augmented Lagrangian Type Penalty Function. *Mathematical Programming*, 23(1):75–86, 1982. ISSN 00255610. doi: 10.1007/BF01583780.
- [355] Pedro E. Zadunaisky. A Method for the Estimation of Errors Propagated in the Numerical Solution of a System of Ordinary Differential Equations. In Georgios Ioannou Kontopoulos, editor, *The Theory of Orbits in the Solar System and in Stellar Systems. Proceedings from Symposium no. 25 held in Thessaloniki, August 17-22, 1964*, page 281. Academic Press, London, 1964.
- [356] Pedro E. Zadunaisky. On the Accuracy in the Numerical Solution of the N-Body Problem. *Celestial Mechanics*, 20:209–230, 1979. doi: 10.1007/BF01371363.
- [357] Federico Zuiani, Massimiliano Vasile, Alessandro Palmas, and Giulio Avanzini. Direct transcription of low-thrust trajectories with finite trajectory elements. *Acta Astronautica*, 72:108–120, mar 2012. ISSN 00945765. doi: 10.1016/j.actaastro.2011.09.011.

Vita

Etienne Pellegrini was born in Strasbourg, France. He graduated in November 2011 with a French “Diplome d’Ingenieur” from Supaero, the leading European Aerospace Engineering school. During a gap year in 2009-2010, he gained invaluable experience interning for Astrium Space Transportation, developing and implementing hazard avoidance and trajectory determination techniques. As part of his French degree, he also had the opportunity to intern abroad: in the Spring of 2011, he worked on trajectory optimization with Dr. Ryan P. Russell at the Georgia Institute of Technology, before moving to Mountain View, California, where he implemented dust lifting mechanisms in NASA Ames’ Mars Global Circulation Model. In January 2012, Etienne followed Dr. Russell to Austin, Texas, where he started his Ph.D. program in Aerospace Engineering under Dr. Russell’s supervision, with funding from Astrium Space Transportation. He obtained his M.Sc. from the University of Texas in May 2014.

Permanent address: etienne.pellegrini-at-gmail.com

This dissertation was typeset with \LaTeX^{\dagger} by the author.

[†] \LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth’s $\text{T}_{\text{E}}\text{X}$ Program.